Universidade Fernando Pessoa

Modelo de Engenharia de Software para o Desenvolvimento de Jogos e Simulações Interactivas

Carlos Eduardo Lé Velasquez

Porto, 29 de Dezembro de 2009

Universidade Fernando Pessoa

Modelo de Engenharia de Software para o Desenvolvimento de Jogos e Simulações Interactivas

Carlos Eduardo Lé Velasquez

Porto, 29 de Dezembro de 2009

Carlos Eduardo Lé Velasquez

Modelo de Engenharia de Software para o Desenvolvimento de Jogos e Simulações Interactivas

Orientador: Professor Dr. José Ângelo Braga de Vasconcelos

Dissertação apresentada à Universidade Fernando Pessoa como parte dos requisitos para obtenção do grau de Mestre em Computação Móvel.

Resumo

Portugal demonstra uma grande debilidade na indústria dos jogos de computador, pois detém poucas empresas dedicadas a este ramo. Assim, pergunta-se qual será a razão para este facto. A obtenção da resposta vai ao encontro do objectivo desta dissertação, sendo que faltam mais cursos dedicados ao ramo e, também, uma metodologia para ajudar ao desenvolvimento dos jogos. Recentemente foram criados dois cursos de desenvolvimento de jogos em Portugal, não sendo este número de todo estimulante. Por outro lado, existem infindáveis metodologias para a criação de Software, mas não existe uma mais específica, exclusivamente dedicada ao desenvolvimento de jogos. Esta dissertação apresentará uma possível metodologia de engenharia de software para o desenvolvimento de jogos de computador e fará um paralelismo entre jogos de computador e simulações interactivas, demonstrando que na realidade são duas áreas idênticas. O estudo empírico pretende demonstrar o interesse dos alunos do ensino superior do curso de Engenharia Informática pelo desenvolvimento de jogos de computador. A criação de uma metodologia de engenharia de software para o desenvolvimento de jogos será apenas um apoio para um curso da área e, caso se comprove exequível, este não perderá nenhuma das vertentes de interesse do curso de engenharia informática, promovendo o estímulo dos seus estudantes. Deste modo, esta metodologia torna-se uma ferramenta, que visa estimular a comunidade académica para esta realidade.

Abstract

Portugal shows a great weakness in the industry of computer games, it has few companies dedicated to this industry. So, the question arises what is the reason for their absence. Obtaining the response is consistent with the objective of this thesis, being that we are missing more courses devoted to business and also a methodology to help the development of games. They just created two courses in game development in Portugal, but this number is not at all exciting. On the other hand, there are countless methods for creating software, but there's not a more specific, exclusively dedicated to game development. This paper will present one possible method of software engineering for the development of computer games and makes a parallel between computer games and interactive simulations, showing that, in reality are two similar areas. The empirical study pretends to demonstrate the interest of students for higher education courses in the Engineering development of computer games. The creation of a methodology of software engineering for game development is only a support for a course in the area, if it is proved feasible, it will not lose any of the aspects of interest to the course of computer engineering, promoting the stimulation of their students. Thus, this approach becomes a tool to stimulate the academic community to this reality.

Synthèse

Portugal présente encore une débilité en ce qui concerne l`industrie des jeux de l`ordinateur, car il y a peu d`entreprises qui travaillent cette branche de connaissances. Donc, on demande quelle sera la cause de ce fait. La réponse à cette question, nous emmène à l`objectif de cette dissertation, car il nous manque encore plus de cours qui s`en occupent et il faut, aussi, une méthodologie qui aide le développement des jeux. Récemment, au Portugal, on a crée deux cours pour le développement de ces jeux, mais ce nombre n`est pas suffisant pour répondre aux besoins des jeunes intéressés. Cependant, il y a beaucoup de méthodologies pour la création de Software, mais il n`y a aucune plus spécifique qui s`occupe, exclusivement, du développement de ces jeux. Alors, cette dissertation présente une méthodologie possible du côté de l`Ingénierie de Software pour le développement des jeux de l`ordinateur. Elle fera un parallélisme entre les jeux de l`ordinateur et les simulations interactives en mettant en évidence qu`il s`agit de deux champs identiques. Cet étude empirique veut, aussi, démontrer l´intérêt des étudiants de l`enseignement supérieur du cours de l `Ingénierie Informatique par ce thème. La création d´une méthodologie de l`Ingénierie de Software pour le développement des jeux sera seulement un appui pour un cours dans ce domaine et, s`il est exécutable, il ne perdra pas un des points d`intérêt du cours de L`Ingénierie Informatique, en promouvant, simultanément, le stimule de ses étudiants. Par conséquent, cette méthodologie est vraiment importante puisqu` elle vise à l`encouragement de la communauté académique pour cette réalité.

Ao meu orientador, Professor Doutor José Ângelo Braga de Vasconcelos, um agradecimento especial por toda a dedicação e esforço demonstrados durante a elaboração deste trabalho, especialmente pela paciência.

Ao Professor Doutor Nuno Jorge Gonçalves de Magalhães Ribeiro, pelo apoio demonstrado e a divulgação do questionário aos alunos de Engenharia Informática da Universidade Fernando Pessoa.

Ao Professor Doutor Rui Leandro Alves da Costa Maia e ao ES-CEFOC, pelo apoio e o espaço online utilizado para a realização do questionário.

A todos os restantes professores do curso de Engenharia Informática da Universidade Fernando Pessoa.

À minha namorada Estela Maria Pinheiro Enxurreira por todas as horas perdidas a ajudar-me.

Aos meus pais Gabriel Pedro Dias Velasques e Maria João Vilaça Lé e à esposa do meu pai, quem eu considero como uma segunda mãe, Maria de Fátima Lopes Velasques, por toda a ajuda que me prestaram durante a elaboração do trabalho.

Ao meu irmão e esposa, João Pedro Lé Velasquez e Ana Filipa da Costa Pinto pela ajuda prestada na revisão do texto

Aos pais da minha namorada, Adriano Manuel da Graça Enxurreira e Orquídea Maria da Silva Pinheiro por toda a paciência que demonstraram.

Aos meus amigos e colegas pela ajuda e apoio que tem vindo a demonstrar durante toda a minha vida académica.

Obrigado

Índice

Índice de Gráficos

Índice de Figuras

Índice de Tabelas

Índice de Esquemas

1. Introdução

O objecto primário de um jogo de computador, é criar uma quantidade de estruturas fundamentais de fácil utilização, para que haja um interface entre humano e a máquina o mais natural possível.

Muita gente sabe quando é que teve o primeiro computador, mas poucos sabem desde quando se tornaram bons utilizadores, dessas máquinas que hoje em dia inundam o mercado de trabalho, empresas e casas.

Se a informática apesar de ser supostamente uma área complexa, tornou-se, hoje em dia, imprescindível para a nossa vida, como é que as pessoas tiveram e têm tanta facilidade em utilizá-la? Para isto contribuiu a tarefa que tanto diverte fazer e refazer uma infinidade de vezes! "O Jogo". Jogar em todo o tipo de hardware que tenha sido considerado topo de gama na altura. Do ponto de vista do utilizador, é um óptimo passatempo e ajuda a adquirir uma capacidade de interacção superior entre humano e computador pessoal (PC).

No caso do programador ou equipa de programação de jogos, tudo o que foi alcançado no ponto de vista do utilizador é apenas o resultado de todo um imenso processo de imaginação, de criação, trabalho e muito mais. Acima de tudo, a construção de um jogo é o seu mundo real, emprego e sustento; mas na sua perspectiva, um jogo é uma visão do mundo "a sua", como ele é, como ele acha que é, ou até como ele gostaria que o mundo fosse. Depois dos sistemas operativos e da inteligência artificial a criação de jogos é actualmente para os programadores o último desafio, mas que ao contrário da programação anterior esta exige do hardware uma evolução ainda mais rápida, com o ultrapassar de limites antes inimagináveis. O mundo em que vivemos é uma realidade, o mundo dos jogos é uma realidade virtual – cabe ao programador imaginar, estruturar, construir o jogo, cabe ao utilizador/jogador utilizá-lo nos seus limites. Assim um utilizador/jogador joga para se divertir e um programador/jogador joga para trabalhar.

Mas isso não é tudo, os jogos são, hoje em dia, um impulsionador titânico de toda a indústria informática, tanto a nível de hardware, bem como de novas técnicas de programação de software e novas linguagens, que podem ser utilizadas em aplicações

de outras áreas como na engenharia, na simulação de barragens, pontes e edifícios, na arquitectura, com a simulação animada de ambientes sociais e de espaços construídos, na medicina na simulação de operações cirúrgicas e até no apoio a essas mesmas operações.

Tendo em conta que o tempo que um sistema operativo, como por exemplo o Windows XP, levou para dar a vez ao Windows Vista (cerca de 7 anos), se a indústria de hardware acompanhasse em demora a criação dos sistemas operativos, só agora é que estariam a ser criadas novos dispositivos de memória de computadores (RAM), Motherboards e disco rígidos, pois não havia necessidade de evoluir muito para além dos processadores anteriores. Foram também jogos, com a sua constante exigência e apetite de novas tecnologias e ferramentas para atingirem novos patamares de qualidade que empurraram a fasquia das peças físicas para níveis em tempo imprevisíveis. Orientando assim os consumidores assíduos de jogos de computador a adquirirem sem hesitar essas novas peças, necessárias para experimentar os jogos de modo a atingir os limites que eles permitem. Neste sentido os jogos de computador definem muitas e variadas fontes de rendimento e de evolução para toda a indústria informática.

Deste modo se é assim tão óbvio que as mais-valias, as vantagens e a percentagem de lucro dos jogos são tão elevadas, porque é que há tão pouco investimento nacional nesta área da indústria? O problema é que o desenvolvimento da programação de jogos não foi considerada como prioritária e como tal não havia uma aprendizagem especializada para o efeito até há bem pouco tempo e mesmo agora é insuficiente.

Com esta breve introdução espera-se afastar alguns preconceitos que existem sobre os jogos de computador, as suas aplicações, não só lúdicas, e como esta área da informática tem tido um papel muito importante e quiçá decisivo no desenvolvimento do software de todas as áreas do saber. Se para uns o jogo virtual é o limite, para outros, da microcirurgia à conquista do espaço, o limite é a imaginação.

1.1. Definição do Problema

O principal problema reside no ensino das tecnologias informáticas, por um lado por não se achar necessário e por outro porque não existe espaço num curso superior para leccionar programação de jogos. Assim o que é necessário é um modelo de ensino

baseado no já utilizado mas orientado para o desenvolvimento de jogos. Algumas das cadeiras mais específicas do curso de Engenharia Informática podem facilmente ser adapatadas de modo a introduzir conceitos de programação de jogos no seu programa. O modelo de Engenharia de Software existente foi criado de modo a ajudar ao desenvolvimento de todo o tipo de software, mas se um jogo de computador também é um software, porque não juntar os dois.

1.2. Objectivos do Estudo

O principal objectivo desta dissertação será apresentar um modelo de engenharia de software para o desenvolvimento de jogos de computador e simulações interactivas. Apesar de serem duas áreas diferentes do desenvolvimento de software ambas dependem de ambientes de trabalho idênticos. É necessário reparar que uma simulação interactiva de uma casa, depende de um motor de gráficos, por exemplo 3D, tal como um jogo, assim um modelo de desenvolvimento dedicado aos jogos poderá facilmente ser usado para as simulações interactivas. Tal como também será demonstrado neste texto, a interactividade é o que faz do jogo de computador um jogo, sem esta não seria possível interagir com as personagens, objectos e ambientes, logo sem interactividade não existiria jogo.

Com estes conceitos em mente será desenvolvido um modelo de possível utilização para a criação de ambos os casos, demonstrando a relevância que o ensino dedicado para esta área seria uma boa hipótese tanto para os alunos como possivelmente para a indústria de jogos.

1.3. A Relevância da Investigação

O ensino utilizando jogos de computadores já é uma realidade. É possível, por exemplo, ensinar, a um baixo nível, a contar utilizando jogos ou, a um nível mais elevado, ensinar a programar ou fazer cálculos matemáticos. Tudo dependeria do envolvimento do jogador com o jogo e a vontade que o jogo incutiria no jogador de chegar mais longe.

Um dos melhores processos de ensino é manter o aluno interessado em aprender e criar a vontade de ver mais, chegar mais longe, e é a mesma razão que leva um jogador a querer chegar ao fim do um jogo. Desvendar o caminho, resolver o último

enigma. Dependendo do jogo, o jogador aprende sempre alguma coisa, porque não usar esta característica para o ensino?

Actualmente o jogador não aprende só com o jogo, com numa nova abordagem na tecnologia dos jogos, o computador é que aprende com o jogador. Ou seja os jogos em vez de ensinarem o jogador a fazer algo, permitem que seja o jogador a ensinar o jogo. O conceito tem como sigla GAWP (*Games With a Purpose*) que engloba um tipo de jogos que retiram do jogador capacidades que o jogo não consegue obter sozinho. Um exemplo muito simples é o de dar títulos a imagens. Tudo indica que o ensino e os jogos de computadores se podem completar, assim, o que falta para elevar a fasquia é introduzir a criação de jogos no ensino superior.

Um aluno será mais facilmente motivado a seguir um certo ramo se existir alguma componente mais relevante e encontrar desafios mais motivantes no inicio dos seus estudos. Falando com alunos, muitos demonstraram que a motivação que os levou a entrar para o ramo da informática foi a dos jogos de computadores.

Os benefícios do desenvolvimento de jogos vão bem mais longe do que simples ideias populares que jogos são brinquedos. Jogos de computador necessitam da integração de várias técnicas de computação gráfica, interfaces multimédia, redes e inteligência artificial. Como resultado os jogos podem ser a base e a estrutura de projectos mais interessantes para os alunos, aumentando assim o seu empenho no trabalho.

Assim estes atributos sugerem que em quase todos os cursos, os jogos podem ser uma ferramenta poderosa para a criação de profissionais excepcionais.

1.4. A Estrutura da Dissertação

O restante documento está dividido em seis capítulos, nomeadamente, Engenharia de Software, Simulações Visuais Interactivas, Industria dos Jogos de Computador, Engenharia de Software para a Construção de Jogos, Estudo Empírico e por último a Conclusão.

Na secção de Engenharia de Software é abordado o modelo convencional de desenvolvimento de software utilizando principalmente uma referência nesta área, o

Software Engeneering Body of Knowledge [SWEBOK 2004], como guião para o processo. São também apresentadas as tendências da engenharia de software tais como o Agile e extreme programming.

Na secção de Simulações Visuais Interactivas, aborda-se algumas das áreas nas quais as tecnologias de programação de jogos podem e são usadas de modo a ajudar a, por exemplo, ter uma melhor percepção do que o utilizador irá efectuar ou mesmo aprender a fazer algo.

Na secção de Industria dos Jogos de Computador é abordada a história dos jogos e o processo de desenvolvimento de jogos de computador na sua generalidade, pois para cada tipo de jogo existam algumas diferenças.

Na secção de Engenharia de Software para a Construção de Jogos será abordada a criação de um modelo que fundirá o modelo geral de engenharia de software com o processo necessário para a criação de jogos, criando deste modo um modelo de engenharia de software para simulações interactivas / jogos.

Na secção de Estudo Empírico serão abordados os questionários efectuados no âmbito desta dissertação, de modo a considerar se este tipo de modelo é necessário e se existe interesse pelo lado dos alunos a seguir esta abordagem de ensino em cursos superiores na área da informática.

Na secção de Conclusão são descritos os resultados obtidos, os objectivos cumpridos, a fiabilidade do modelo e as suas limitações. O trabalho futuro a desenvolver também será abordado nesta secção.

1.5. A Metodologia da Dissertação

A metodologia da dissertação compreende uma revisão e análise da literatura nas áreas de engenharia de software e indústrias de jogos. Esta revisão e análise estarão inseridas nos capítulos 2 - Engenharia de Software, 3 - Simulações Visuais Interactivas e 4 - Industria dos Jogos de Computador. Com base nestes pressupostos principais de Engenharia de Software e nos requisitos gerais de programação de um jogo de computador, é desenvolvido um modelo de engenharia de software para o desenvolvimento de jogos de computador que será apresentado no capítulo 5.

O capítulo 6 apresenta o desenvolvimento de um estudo empírico sobre o interesse dos alunos em relação à inserção de programação de jogos no plano curricular de um curso na área da informática, recorrendo a um questionário que foi efectuado via internet aos alunos do Curso Superior de Engenharia Informática de Universidade Fernando Pessoa. Neste capítulo estarão também descritos os resultados e as conclusões obtidas do estudo empírico. O capítulo 7 conclui a dissertação apresentando as conclusões retiradas do trabalho efectuado.

1.6. As Limitações do Estudo

Para uma melhor presença da opinião dos alunos do ensino superior no ramo da informática poder-se-ia ter introduzido o questionário noutros ambientes fora da Universidade Fernando Pessoa, mas o mesmo não foi possível. Poder-se-ia também procurar levar o mesmo questionário a alunos do ensino secundário, principalmente do 12º ano de escolaridade, de modo a perceber exactamente se a programação de jogos afectaria a escolha destes no curso a seguir.

2. Engenharia de Software

2.1. Introdução

A Engenharia é uma ciência que aplica o conhecimento das ciências naturais e da matemática, em conjunto com a experiência adquirida e com as práticas desenvolvidas em projectos reais. O objectivo principal é a criação de benefícios para indivíduos e organizações através de uma utilização eficaz e eficiente de recursos humanos e físicos. A Engenharia de Software segue a mesma linha de orientação: definição, desenvolvimento e manutenção de software útil para as organizações tendo em conta uma gestão dinâmica dos recursos de hardware, software, humanos e financeiros. A visão clássica da disciplina de engenharia de software no âmbito das ciências da computação visa o desenvolvimento e a manutenção do ciclo de vida do software. Esta disciplina estabelece e utiliza sólidos princípios de engenharia para obter um software fiável, o mais económico possível e que funcione correctamente em máquinas reais. A Engenharia de Software estuda os métodos, técnicas, procedimentos e ferramentas que permitem representar, especificar, desenvolver e manter aplicações de software associadas aos diferentes domínios do nosso conhecimento.

2.2. Visão clássica da Engenharia de Software

A metodologia clássica de engenharia de software engloba um conjunto de tarefas a desempenhar durante o ciclo de vida do software. Os diferentes métodos utilizados definem a forma como construir o software e estão associados às diversas fases de desenvolvimento e manutenção:

a) Requisitos de Software

b) Design de Software

c) Construção de Software

d) Testes de Software

d) Manutenção de Software

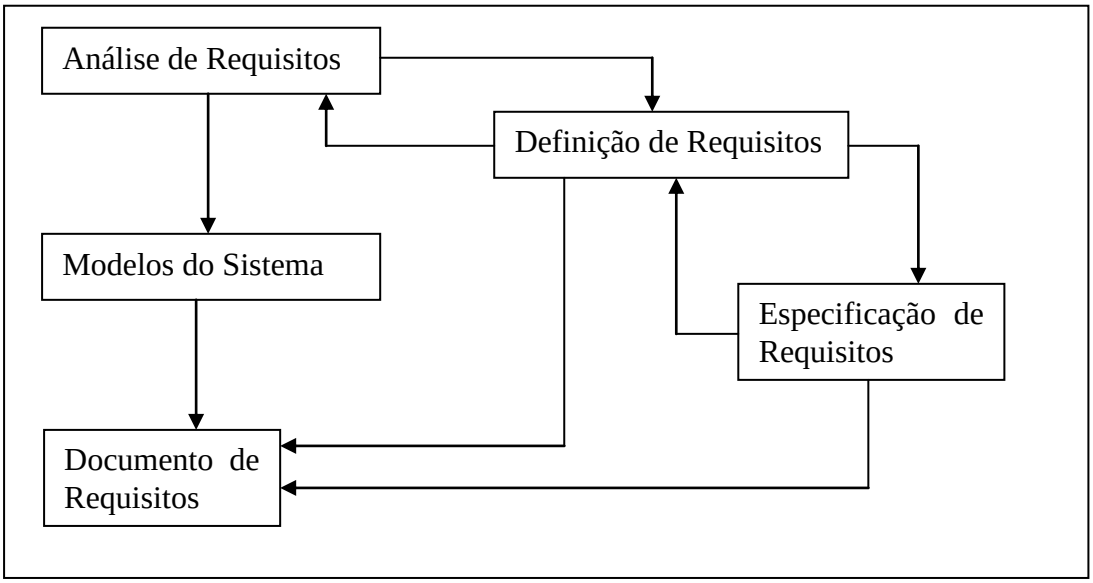
O ciclo de vida de um projecto de software deve contemplar este conjunto de fases e um respectivo conjunto de resultados sob a forma de documentação, protótipos

de software ou simplesmente artefactos de software que serão apresentados gradualmente à entidade (normalmente designada por cliente) a que se destina o projecto de desenvolvimento de software. Estas fases são explicadas em mais detalhe a seguir. A visão clássica de Engenharia de Software segue principalmente o modelo Cascata, dividindo o processo completo em fase, onde para se começar a próxima tem de primeiro se terminar a anterior. Este modelo é explicado mais detalhadamente neste capítulo.

2.3. Engenharia de Requisitos de Software

A análise e recolha de requisitos de software deu origem a uma área nova na engenharia, actualmente descrita por Engenharia de Requisitos. Os requisitos de software exprimem as necessidades e limitações do produto de software que contribuem para a solução do mesmo problema do mundo real. A área de conhecimento dos requisitos de software está preocupada com a análise, especificação e validação de requisitos de software. Caso estas actividades sejam executadas deficientemente os projectos de software ficam criticamente vulneráveis.

O processo de Engenharia de Requisitos é um conjunto de actividades que leva à produção da definição e especificação de requisitos [Sommerville 2004]. Estas actividades, quando completas, dão origem ao documento de requisitos. Este documento define assim o problema que será resolvido pelo novo software.



Esquema 1 Processo de Engenharia de Requisitos (adaptado de Sommerville 2004)

O documento de requisitos de software contem os requisitos do sistema, assim é o documento oficial, necessário para os programadores do sistema. O documento de requisitos inclui a definição e a especificação de todos os requisitos, mas este documento só deverá demonstrar o que o software deve fazer e não como o deve fazer. Deste modo poder-se-á espelhar cada requisito para a parte de design de software que use o mesmo requisito.

2.3.1. Requisitos de Software

A definição de requisitos de software é uma descrição abstracta dos serviços que o software deverá providenciar e as restrições sobre as quais deve funcionar [Sommerville 2004]. Isto significa que todos os serviços, funções e operações que o software terá serão aqui descritos em linguagem natural e diagramas intuitivos. Um requisito é uma afirmação que estabelece como o software, que vai ser criado, se deve comportar. Uma característica essencial é que o requisito de software seja verificável. Por outras palavras, depois de um requisito de software estar implementado, o software tem de se comportar de acordo com as condições estabelecidas no requisito, ou seja no mais básico possível um requisito de software é uma propriedade que tem de ser demonstrada para resolver algum problema do mundo real.

Durante esta fase existem dois tipos de requisitos, funcionais e não funcionais. Requisitos Funcionais descrevem funções que o software executa. Por exemplo a formatação de algum texto ou operação de sistema. Requisitos não Funcionais são os que agem para restringir a solução, ou seja os requisitos que se aplicam aos standards ou qualidade de performance para restringir o design da operação do sistema. Os Requisitos não Funcionais ainda devem ser divididos em mais uma serie de requisitos. Neste caso existem os requisitos do produto, do processo e requisitos externos. Os requisitos do produto representam as necessidades que o software terá de cumprir, tipo velocidades de transferência de dados ou cálculos. Os requisitos do processo são datas de entregam standards necessários e os requisitos externos englobam requisitos administrativos das organizações ou necessidades éticas.

A especificação de requisitos adiciona mais informação à definição de requisitos, assim introduzindo o mais ínfimo pormenor de cada requisito. Por causa deste nível de detalhe o uso de uma linguagem natural, pode não ser o método mais

eficaz e deste modo é necessário encontrar alternativas para descrever estes requisitos. Algumas alternativas incluem notações gráficas, especificações matemáticas e outras.

2.3.2. Identificação, análise, e definição de requisitos

Se o projecto for viável, então torna-se necessário planear o seu desenvolvimento. É preciso, nesta fase, definir a equipa de desenvolvimento, as actividades necessárias, atribuir pessoas às actividades e estimar o tempo para cada actividade bem como para o projecto. Nesta fase tem que existir uma boa comunicação entre a empresa e o cliente de modo a que todos os requisitos do sistema sejam bem entendidos. Esta fase deverá estabelecer os requisitos do sistema de que o software irá fazer parte, ou seja, a identificação do conjunto de requisitos que satisfazem o novo sistema de software

Em qualquer organização existem processos de troca e partilha de fluxos de informação que precisam de ser bem analisados pela equipa de desenvolvimento do Software. Em função dos requisitos identificados anteriormente, é necessário desenvolver um trabalho de análise nos domínios de informação a representar, funções pretendidas, desempenho e interface com o utilizador.

Uma vez efectuada a análise a equipa pode então passar ao próximo passo, trata-se agora de elaborar o design de software. Ao contrário do que se possa pensar o design não implica apenas a apresentação do ambiente do sistema, ele inclui principalmente a arquitectura de como o software se comportará.

2.4. Design do Software

Com a crescente complexidade dos sistemas de software a necessidade da fase de design de software torna-se ainda mais evidente. Esta complexidade implica um risco acrescido do sistema não alcançar os seus objectivos. Para se evitar este risco é necessário uma plano específico para o desenvolvimento do software. O design de Software facilita assim a criação do software fornecendo um guia e possibilita as primeiras avaliações da aplicação

Como está descrito no Software Engineering Body of Knowledge, "*Design is defined as both the process of defining the architecture, components, interfaces and*

other characteristics of a system and the result of [that] process", o design é definido como tanto o processo de definir a arquitectura, os componentes, os interfaces e outras características do sistema e do resultado desse processo.

O Design de software demonstra uma papel importante no desenvolvimento de software permitindo, a equipa, desenvolver modelos e planos que constituirão no futuro a solução a ser desenvolvida. Esta fase envolve o desenvolvimento de Algoritmos e Estruturas de dados, Arquitectura de software, Regras e Procedimentos. Este conjunto deve traduzir os requisitos de software em desenvolvimento, cuja qualidade deve ser aferida antes da codificação, assim deve ser bem documentada, fazendo parte da configuração do software.

O Design de Software engloba, tal como os Requisitos de Software, uma série de fases. Assim numa primeira fase é necessário resolver alguns problemas chave do sistema, problemas que não englobam a programação mas sim objectivos, restrições e alternativas de que dispõe. Após isto, numa nova fase, serão planeados os eventos do sistema e como estes interagem, a distribuição de componentes e manuseamento de erros e excepções. Uma vez obtidas estas fases inicia-se o processo de desenhar a arquitectura do sistema, ou seja estruturas, estilos e escolhas das linguagens de programação. Com isto executado terá de preparar os atributos de qualidade e técnicas de avaliação da mesma.

Para finalizar existem duas últimas fases, uma de descrições do design tanto estáticas ou estruturais e dinâmica ou dos comportamentos do sistema e a última a fase de planeamento de estratégias em geral, orientadas a funções, orientadas a objectos e orientadas a estruturas de dados.

2.4.1. Arquitectura de Software

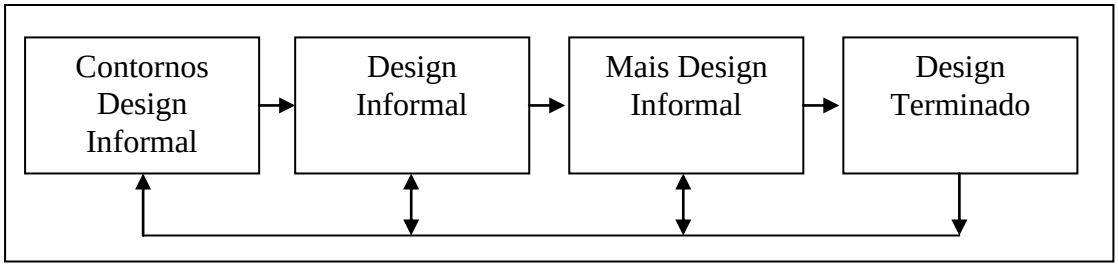
A Arquitectura de software é dividida em duas fases, a arquitectura genérica e arquitectura detalhada, estas também podem ser, de acordo com o software engineering body of knowledge, descritas como design de alto nível e design detalhado. A arquitectura genérica é a fase o software e decomposto e organizado em módulos e nas suas relações. Ou seja descreve a organização fundamental do sistema de software, identificando os módulos e as relações entre cada um e o ambiente, para que alcancem os objectos pretendidos.

Já a arquitectura detalhada, preocupa-se com a descrição detalhada de cada módulo, tornando possível o desenvolvimento adequado.

2.4.2. Processo do Design de Software

Um modelo genérico do design de software é um simples gráfico direccionado para ao software. O objectivo é atingir a criação deste gráfico sem inconsistências. Para este gráfico ser concluído são necessárias várias iterações de modo a obter o melhor gráfico possível. O processo de design envolve então vários modelos do software com diferentes níveis de abstracção, oferecendo assim a possibilidade de melhorar modelos anteriores. Obtendo assim um progresso do processo de design simplificado mas eficaz.

À medida que o progresso dos modelos vai sendo realizado o detalhe de cada modelo vai aumentando igualmente. Deste modo obtém-se um modelo final que oferece especificações precisas dos algoritmos e estruturas de dados necessárias para a implementação.



Esquema 2 Processo do Design de Software (adaptado de Sommerville 2004)

2.5. Construção do Software

A fase de construção de software refere-se a todo o processo de criação, através da codificação, unidades de teste, integração e debugging. Nesta fase será desenvolvido todo o sistema utilizando os recursos e design apresentados nas fases anteriores.

2.5.1. Noções Fundamentais

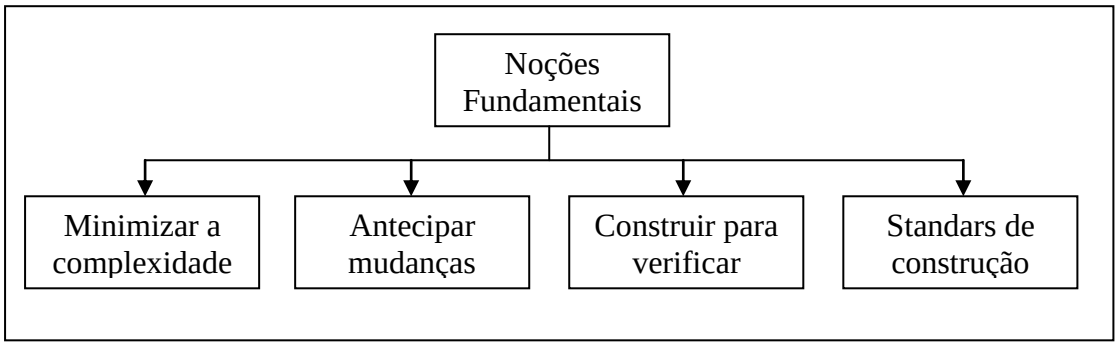
Existem quatro noções fundamentais para a construção de software que englobam, minimizar a complexidade, antecipar mudanças, construir para verificar e standards de construção. A necessidade de minimizar ou reduzir a complexidade é

simples de compreender, pois à medida que o desenvolvimento avança a complexidade aumenta e se não houver cuidado pode ficar impossível de compreender a programação.

Antecipar mudanças segue o exemplo da noção anterior pois se durante o desenvolvimento já se estiver à espera que a qualquer instante tenha de se alterar alguma parte do código será mais fácil de o fazer, para tal existem varias técnicas que ajudam as eventuais alterações.

Construir para verificar significa exactamente isso, se já se desenvolveu software de maneira a que a falhas sejam facilmente detectadas e corrigidas tanto pela equipa que está a desenvolver como na parte de testes do sistema acelera o processo e também simplifica.

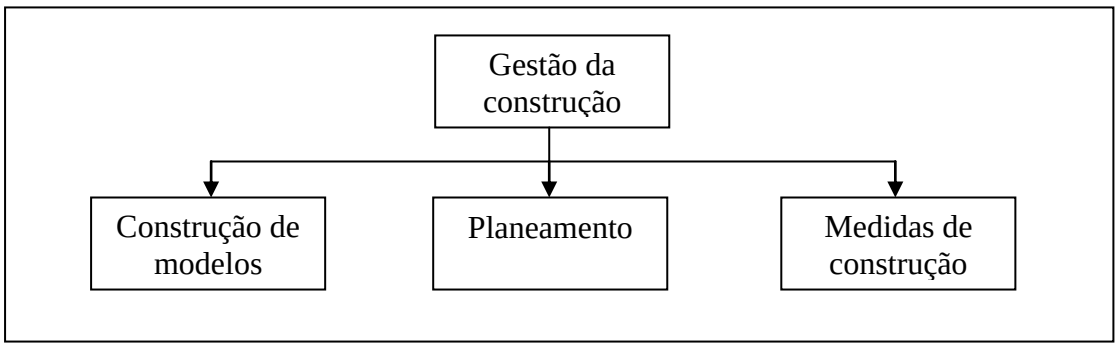
Os Standards de construção são as aplicações que directamente afectam o desenvolvimento. Nestes existem quatro grupos principais, as linguagens de programação, métodos de comunicação, as plataformas de desenvolvimento e as ferramentas usadas para ajudar. As linguagens de programação serão, por exemplo, o C ou C++, os métodos de comunicação usados serão a maneira utilizada para estabelecer uma comunicação entre os elementos da equipa, por exemplo o tipo de formato de texto a ser usado, DOC ou PDF. As plataformas serão os sistemas para os quais se está a desenvolver. E para acabar as ferramentas de ajuda serão aplicações de apoio tipo o UML (Unified Modeling Language).



Esquema 3 Noções fundamentais – Construção de Software (adaptado de Sommerville 2004)

2.5.2. Gestão da Construção de Software

No intuito de gerir o processo de construção ter-se-á de usar modelos de construção, planear e calendarizar os eventos ou actividades. Existem vários modelos de construção de software, entre eles uns que são mais lineares que outros mas todos têm o objectivo de apresentar um ciclo de vida do desenvolvimento do software. O modelo Waterfall, Ágil (SCRUM), Stage Gate e as técnicas RAD (Rapid Application Development) são alguns dos modelos mais conhecidos que serão abordados posteriormente. Todas estas metodologias podem ser usadas, dependendo do rigor e do trabalho que está a ser realizado.



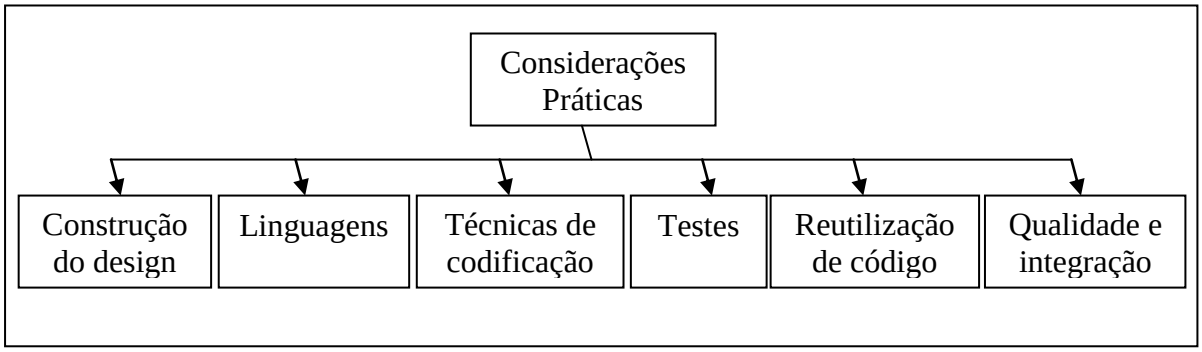
Esquema 4 Gestão - Construção de Software (adaptado de Sommerville 2004)

A escolha do método é um aspecto essencial para a construção mas também se tem de planear a sequência do que será desenvolvido, por exemplo não se pode criar design gráfico sem possuir os componentes visuais. Uma vez criado, o plano terá de se calendarizar cada passo de modo a ter uma data aproximada para terminar o desenvolvimento.

2.5.3. Considerações Práticas

No processo de codificação existem sempre algumas considerações que todos os elementos de uma equipa de programação precisam de pensar. A construção do design, as linguagens utilizadas, as técnicas de codificação, os testes, a reutilização de código, a qualidade e a integração. Todos estes pontos tem de ser considerados para que a equipa possa mais facilmente se entender e assim prosseguir mais rapidamente. Por exemplo se nas técnicas de codificação o programador deve criar código compreensível, usando nomes para as variáveis perceptíveis, comentar o código e gerir os próprios erros.

Todas estas considerações são necessárias para qualquer equipa de desenvolvimento de software, só quando se fizerem estas práticas é que poderá ser considerada uma verdadeira equipa.



Esquema 5 Considerações Práticas – Construção de Software (adaptado de Sommerville 2004)

2.6. Testes do Software

Teste de software consiste na verificação dinâmica do comportamento do programa num conjunto de testes finitos. Aplicação de mecanismos que permitam detectar eventuais erros cometidos na fase de análise ou na fase de concepção. Os testes podem ser de variada ordem, tais como: testes associados à lógica interna do sistema de software, onde são efectuados testes às instruções; ou validações ao funcionamento externo, em que se verifica se determinados valores de entrada produzem os resultados esperados. É necessário ter em conta o software que está a ser desenvolvido, no caso de pequenos programas, os testes são simples e fáceis de executar, mas no caso de grandes sistemas, os testes tem de ser executados por fases. O processo de teste dever ser então feito incrementalmente juntamente com a implementação do software. Geralmente existem 5 fases, os testes de unidades (unit testing), testes de módulos (module testing), testes de subsistemas (sub-systems testing), testes de sistema (systems testing) e testes de aceitação (Acceptence testing).

Na fase de unit testing são testados componentes individualmente, garantindo que funcionam independentemente. Na module testing são testados vários componentes dependentes de outros, tipo classes de objecto, funções e procedimentos. A fase de sub-system testing envolve o teste dos módulos integrados em partes integrais do sistema. System testing testa a integração de todos os sub-systems de modo a encontrar erros nas

interacções dos mesmos sub-systems. E a Acceptance testing são os testes finais executados já com dados oficiais, de modo a testar se o sistema realiza as tarefas pretendidas.

Para além do processo de testes genérico também são hoje em dia utilizadas técnicas de desenvolvimento orientadas aos testes (Test Driven Development). Estas técnicas oferecem a possibilidade de testar secções de código sem a necessidade de executar todo o código, assim diminuindo o tempo de debug e compilação. Na realidade o programador primeiro necessita criar as unidades de teste automáticas que definem os requisitos do código antes de escrever o código em si, deste modo quando o código estiver completo é logo testado para comprovar o seu resultado.

2.7. Manutenção do Software

O software é um produto que tem um tempo de vida útil, sendo assim necessário efectuar periodicamente actualizações. Consequentemente as organizações necessitam de actualizar periodicamente o seu sistema de software, o seu hardware, assim como desenvolver programas de formação dos respectivos utilizadores do sistema. Em termos globais o processo de manutenção é motivado por alterações no ambiente exterior (novas versões de hardware / software), por alterações solicitadas pelos clientes que pretendem novas funções ou melhor desempenho do software, ou por erros detectados no sistema.

Existem 3 tipos de manutenções que podem ser executados após a entrega do software. Primeiro a manutenção correctiva que tal como já foi dito, preocupa-se com os erros detectados no software já em utilização. Estes erros são normalmente conhecidos por bugs do sistema. Segundo é a manutenção adaptativa, que normalmente significa alterar o software para funcionar num novo ambiente, tipo funcionar em sistemas operativos diferentes. O terceiro é a manutenção perfectiva, utilizada para implementar novas funcionalidades ao Software que o cliente possa ter exigido, estas funcionalidades são novos requisitos funcionais e não funcionais.

2.8. Modelos Clássicos

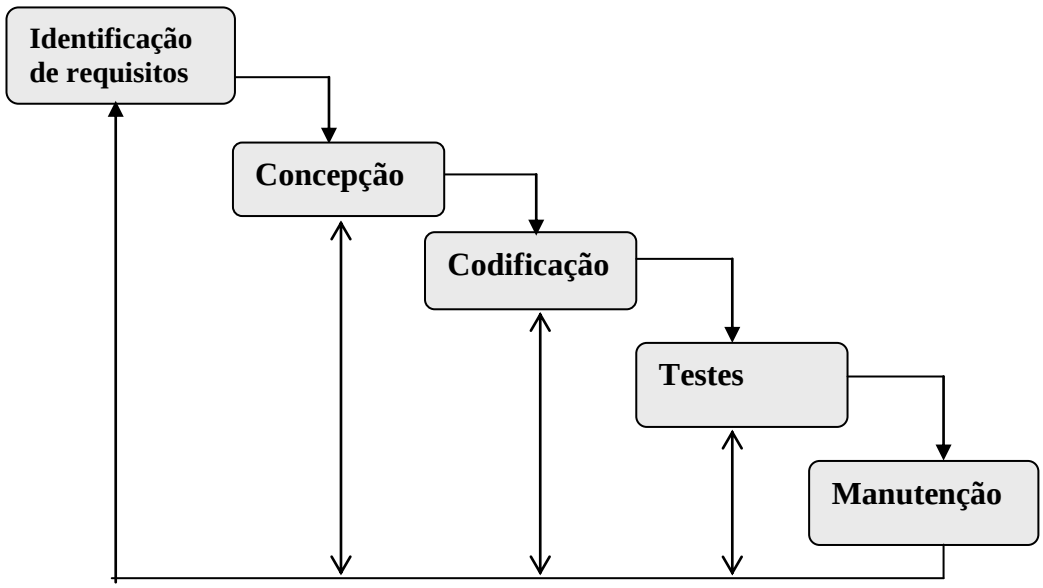
Os modelos para desenvolver o Processo de Software são um conjunto de passos que devem ser tomados durante todo o processo de criação de um software.

Dependendo do que for desenvolvido existem modelos mais adequados do que outros. A seguir apresentar-se-ão alguns dos modelos mais utilizados.

2.8.1. Modelo Cascata (WaterFall)

É um modelo de desenvolvimento de software sequencial, no qual se verifica um fluir constante (como uma cascata) através das fases de análise de requisitos, projecto, implementação, testes (validação), integração, e manutenção de software.

Este método simples não é utilizado em grandes projectos, mas para melhor expor o processo de criação é o mais fácil e rápido de explicar.



Esquema 6 Modelo Waterfall

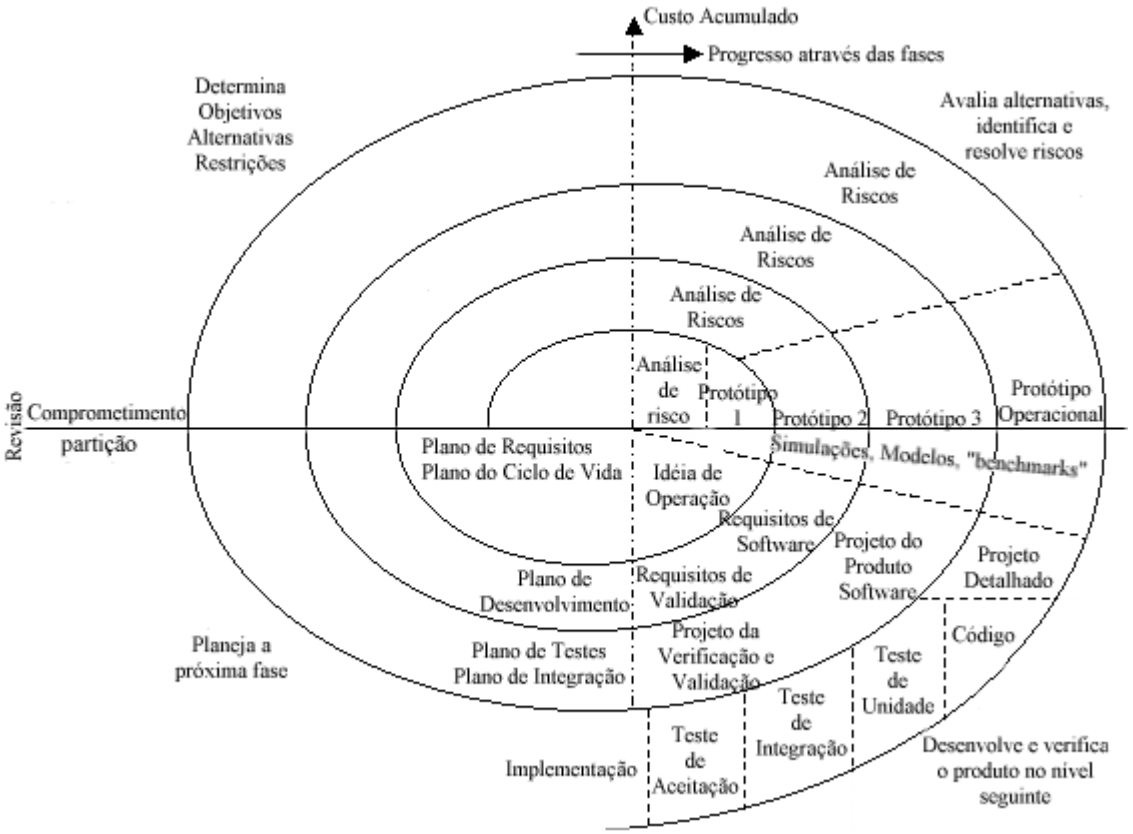
Este modelo é constituído por cinco fases. A fase posterior só é realizada quando a sua antecessora estiver concluída e assim sucessivamente. Se um erro for detectado numa determinada fase, o analista só tem que recuar à fase imediatamente anterior para corrigir esse erro.

2.8.2. Modelo Espiral

O processo de desenvolvimento clássico foi um pouco alterado com este modelo. O modelo espiral, desenvolvido por Boehm, envolve algumas melhorias em relação aos modelos clássicos, mas não os descarta, na realidade estes modelos podem ser utilizados juntamente com o modelo espiral. O modelo tem o nome espiral porque usa os ciclos da espiral para repetir todas as fases que forem necessárias para o

desenvolvimento do software. A titulo de exemplo ir-se-á explicar um ciclo de 4 fases, mas o modelo não esta sujeito a um numero fixe de fases.

As fases englobam determinar objectivos, avaliar alternativas e riscos, desenvolver e verificar e para terminar planear o próximo passo. Como o esquema demonstra cada fase é repetida até o software estar completo. Na primeira fase determina-se o objectivo para o ciclo que vem a seguir, na segunda avalia-se alternativas possíveis e identificam-se e resolvem-se os possíveis riscos, para na terceira fase se poder desenvolver o software. Na quarta fase planeia-se o próximo ciclo, a não ser que o software já esteja completo, caso seja verdade termina-se o desenvolvimento.

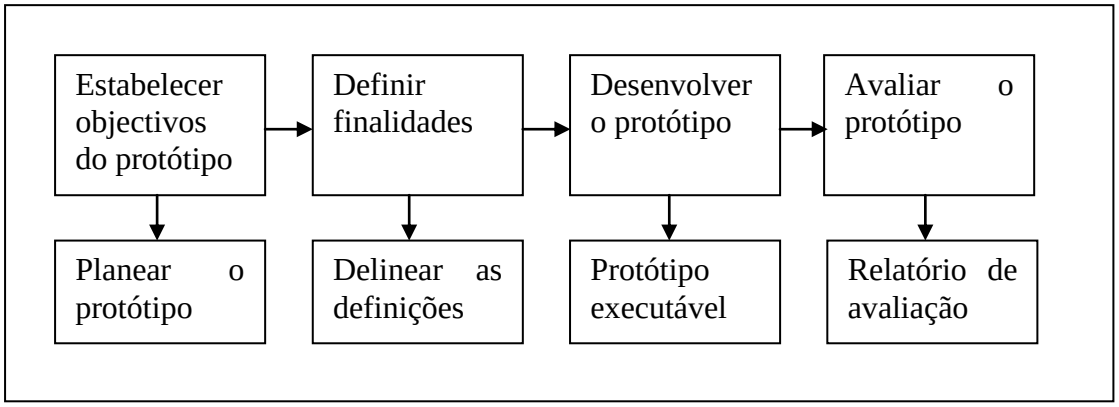


Esquema 7 Modelo Espiral (Boehm 1988)

2.8.3. Prototipagem

Uma das maiores dificuldades para o desenvolvimento de software e a dificuldade que os clientes têm para demonstrar o que desejam que o software faça, para contrariar esta dificuldade foi criado o modelo de prototipagem, que se resume à criação rápida de protótipos e melhorando estes até se atingir o resultado pretendido.

Os benefícios deste modelo englobam melhor compreensão entre cliente e programador à medida que se apresentam os protótipos, serviços em falta podem ser descobertos, dificuldades no uso podem ser identificadas e refinadas, os programadores podem detectar requisitos em falta, um sistema a funcionar esta rapidamente disponível para demonstrações e o protótipo é usado como base para criar as especificações do sistema.

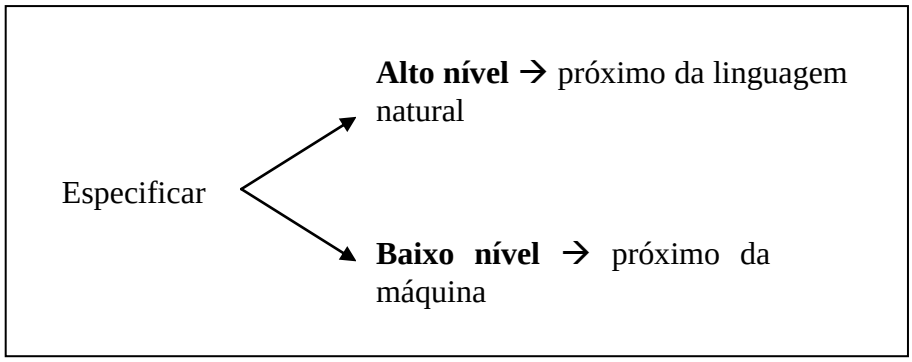


Esquema 8 Processo da Prototipagem (adaptado de Sommerville 2004)

2.9. Modelos Ágeis

2.9.1. Rapid Aplication Development (RAD)

Esta forma de desenvolvimento de software está associada a um conjunto de ferramentas que surgiram no final da década de 80. Estas ferramentas permitem especificar as características do software a alto nível e posteriormente a geração automática de código, ou seja, a ferramenta gera o código fonte de forma automática baseado na especificação efectuada a alto nível. A base deste modelo consiste em especificar software a um nível próximo da linguagem natural.



Esquema 9 Modelo RAD

Este modelo é constituído pelas seguintes fases de desenvolvimento, levantamento de requisitos, definição de uma estratégia para o projecto, implementação com ferramentas, testes, documentação e manutenção

2.9.2. Modelo Ágil (SCRUM)

Este modelo de desenvolvimento oferece métodos para definir o planeamento, os principais papéis (roles) e a forma de trabalho. A ideia do SCRUM e definir papéis bem específicos para cada elemento da equipa de trabalho e o que cada um vai ter de fazer para o projecto ser terminado.

Sinteticamente o método funciona da seguinte maneira:

- O produto é definido: os requisitos e o que o cliente quer.
- Assim o chefe da equipa define as funcionalidades mais importantes e com estas cria uma lista do que será necessário fazer.
- Com as prioridades definidas serão definidos os Sprints, ou seja as funcionalidades são divididas por sprints, que representam o tempo para realizar cada tarefa.
- Cada sprint possui uma das funcionalidades da lista, e devem ser feitos de acordo com as prioridades definidas. Estes Sprints devem ser preparados para durarem de 2 a 4 semanas e no final de cada período, tenham produto para apresentar.
- Os Sprints vão sendo feitos até terminar a lista de tarefas e assim o projecto ficar completo.

Mais detalhadamente o SCRUM possui três partes principais: os Papeis, as Cerimónias e os Artefactos.

Os papéis são o Product Owner, ou seja o proprietário do produto que representa os interesses do cliente. O ScrumMaster, o líder da equipa de desenvolvimento e a equipa em si. As cerimónias são eventos que acontecem dentro do ciclo de desenvolvimento, são a reunião de planeamento do Sprint, reuniões diárias e a reunião de revisão do sprint. Os artefactos são ferramentas básicas para se trabalhar com este modelo, o Product Backlog, o Sprint Backlog e o Burndown Chart. O Product Backlog é a lista de funcionalidades a serem criadas. O Sprint Backlog é a lista de requisitos e prioridades para funcionalidade que estão a desenvolver, ou seja uma lista que especifica os passos necessários para implementar uma funcionalidade do sistema. E o Burndown Chart que é um gráfico que demonstra a quantidade de trabalho cumulativo restante de um sprint.

2.9.3. Stage Gate

É um método que evoluiu do método ágil, acrescentando apenas alguns passos para melhorar o processo de criação, neste caso de jogos, criando uma gestão de recursos (humanos e monetários) melhorada.

O Stage Gate acrescenta então, uma série de portagens (gates) que cada Sprint inicialmente tem de atravessar. Nestas portagens os Sprints são avaliados de acordo com alguns pré-requisitos impostos pela empresa ou chefe. Quando o projecto não é aceite, ou seja não cruza a portagem, todos os recursos atribuídos ao Sprint são transferidos para os restantes que conseguem cruzar a portagem. Cada Sprint que falhe não é necessariamente descartado, todo o material desenvolvido até ao momento é guardado ou reciclado para uso posterior.

Assim, todo o projecto impõe datas e recursos iniciais a cada equipa, enquanto estas tentam desenvolver a melhor funcionalidade possível; caso não consigam convencer na data marcada, todos os recursos são então redistribuídos pelos restantes marcando nova data para uma nova portagem. Finalmente só existirá o projecto final.

2.9.4. Extreme Programming

Este método é uma proposta de desenvolvimento Ágil e iterativo, isto significa que é realizado um desenvolvimento rápido e iterativo de pequenas partes da funcionalidade do software. Estas partes devem ser constantemente melhoradas. Deste modo o modelo oferece uma possibilidade de entrega rápida do código tornando o cliente um membro activo do desenvolvimento, pois é a sua interacção que possibilita o avanço rápido do projecto.

Uma das características importante do Extreme Programming é a não existência de um processo de design tradicional com a elaboração de modelos de arquitectura de software. Assim este modelo não se prende a nenhum conceito pré feito para o desenvolvimento de software, libertando assim as empresas à escolha do método que melhore lhes convir.

3. Simulações Visuais Interactivas

3.1. Introdução

Uma simulação interactiva, na realidade é um jogo, contem um actor, um objectivo e até contem uma história, embora não seja exactamente no mesmo contexto do jogo, mas as simulações têm todas estas características. Na realidade a indústria de jogos de computador também contribui para as simulações, e vice-versa. Para desenvolver um jogo, onde a personagem tem de se movimentar o mais naturalmente possível é necessário saber exactamente como funcionam os músculos, e as suas reacções às forças físicas. Estas informações podem ser facilmente avaliadas com simuladores de movimentos de corpos humanos.

A Simulação interactiva, normalmente mencionada como simulação interactiva visual, pode ser usada nas mais variadas áreas de investigação ou ensino. Uma apresentação de um desenho arquitectónico ou a simulação da inserção de uma agulha cirurgicamente de um paciente, pode ser facilmente transformada num simulação interactiva de modo a ajudar ou ensinar. No primeiro caso o arquitecto poderia simular uma visita à casa e mostrar o aspecto do interior alterando cores, texturas, e objectos ou simplesmente ligando ou desligando uma luz. No segundo caso, que é um caso provado, pode-se visualizar todo o trajecto da agulha cirurgicamente do paciente. [Chentanez, N., et. al. 2009]. Dependendo da área em que está a ser utilizada a simulação interactiva pode ser desenvolvida adequadamente, possibilitando assim melhores ferramentas para cada área.

Pretende-se com este capítulo demonstrar que nas mais variadas áreas cientificas, as tecnologias de simulações interactivas / jogos de computador são efectivamente utilizadas e com resultados bastante produtivos.

3.2. Área da Aeronáutica

Na área da aeronáutica é relativamente simples apresentar um exemplo pois a maior parte dos jogos de aviões, são simuladores de aviação, muito aproximados da realidade. Por exemplo os vários Flight Simulator's que têm vindo a ser desenvolvidos aproximam-se de tal forma da realidade que quase se pode aprender a pilotar um avião apenas por jogar. O Flight Simulator X, a última versão do simulador de voo é o

culminar de 25 anos de jogos, e contem dúzias de aeronaves o mais realistas possível, de modo a conseguir simular vida real. [Flight Simulator X]

Esta área não se baseia apenas em simuladores de aviação que são jogos, pois também os militares e agencias espaciais usam simuladores para treinar os seus pilotos para missões quer dentro ou fora da terra. Por exemplo todos os pilotos espaciais, passam horas nos simuladores para conseguir obter bons resultados e de modo a não falharem quando a verdadeira missão for realizada. Nestes casos já não são apenas os jogos de computador que estão a ser usados, mas simuladores profissionais que para além do simulador por software também usam máquinas de modo a aproximar a experiencia do utilizador à realidade. [Beard, S. et. al. 2008]

3.3. Área da Medicina

A área da Medicina consegue ser tão abrangente quanto a anterior, pois desde simplesmente ensinar os alunos a fazer algumas operações simples até realizar operações cirúrgicas podem ser usadas as simulações interactivas. Um exemplo destas tecnologias a funcionar á a simulação interactiva de inserção de agulhas cirúrgicas e a sua orientação. [Chentanez, N. et. al. 2009]

Com a evolução das tecnologias, até existem simuladores de operações plásticas. Que conseguem se aproximar imenso da realidade, recorrendo a uma fotografia da zona do corpo que se deseja alterar, o programa consegue facilmente distinguir os objectos muito intuitivamente e oferece a possibilidade de simular as alterações pretendidas. Capacidades de simulações de Rinoplastia virtual (plástica no nariz), contorno virtual da face, etc, são oferecidas com o programa. [Cirurgia Plástica Virtual 1.0]

As simulações interactivas são, no seu cerne, um jogo de computador. A primeira vez que uma simulação cirúrgica passou a jogo, foi com o jogo de tabuleiro “Operação” que simulava uma operação a várias partes do corpo humano. Mas com a introdução dos computadores apareceram simulações idênticas com o intuito de ensinar o processo de uma operação, por exemplo, a um joelho. [edHeads]

3.4. Área da Arquitectura

Na área de Arquitectura a simulação interactiva pode ser usada para demonstrar uma obra numa apresentação 3D, aproximando assim o utilizador de uma imagem de como ficará a obra. No exemplo prático do ponto 3.6 deste capítulo, será apresentado um simulador de uma casa em 3D, esta simulação servirá para demonstrar que com o motor de um jogo é possível desenvolver simuladores interactivos dedicados à arquitectura.

Os programas de desenho para arquitectura conseguem apresentar o objecto desenhado num ambiente 3D, mas não oferecem a possibilidade de passear pelo interior da casa interagindo com a mesma, por exemplo ligando e desligando interruptores ou abrindo persianas para ver a luz que entra na casa, simulando a posição do sol.

Neste caso já existem jogos de computador que oferecem a possibilidade de implementar casas e desenhos nos mesmos e depois podendo utilizar os seus desenhos, o Second Life [Second Life] é um exemplo destes jogos. O jogo contém um interface que ajuda a criar objectos e até de os importar de programas de desenho gráfico, assim o jogador poderá usar os seus desenhos e objectos gráficos no jogo.

3.5. Área Militar

A área Militar já usa simulações a vários séculos, desde que os estrategas usavam mapas para planear as próximas batalhas que se pode dizer que realizavam simulações do que desejavam que acontece-se em batalha. Foi só recentemente que estas simulações de estratégia vieram para o mundo dos jogos de computador. Um dos primeiros jogos de estratégia do mundo foi o Dune II que apresentava 3 facções que lutavam entre si pelo domínio de um planeta. A partir deste ícone do mundo dos jogos, os Real Time Strategy Games (RTS) foram considerados um novo tipo de jogo e amplamente desenvolvidos. As simulações de guerra não acabam nos RTS, pois os simuladores de batalhas reais ainda são usados com uso das mais recentes tecnologias de recolha de dados no terreno.

3.6. Exemplo prático de uma simulação 3D

Um exemplo simples mas muito utilizado, são as simulações de construções. É recorrente ver programas de televisão que demonstram simulações de edifícios. Estas simulações são efectuadas em programas de software de desenho do género do Autocad ou 3D Studio. Um dos problemas é a interacção limitada mas utilizando um motor de jogos de computador estas limitações transformam-se em possibilidades. Um motor de jogos consegue abrir um modelo de uma casa, como representam as seguintes imagens, depois com algumas linhas de código pode-se modificar o aspecto, da mesma.

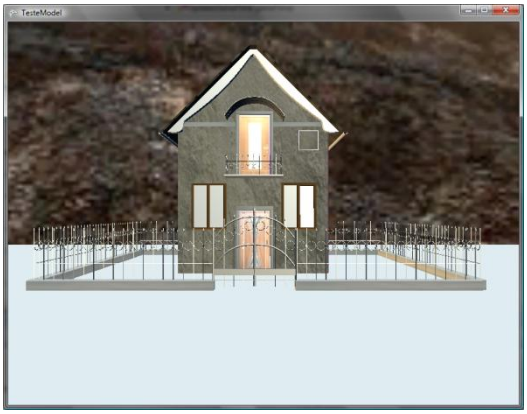


Figura 1 Modelo de uma casa, visão central

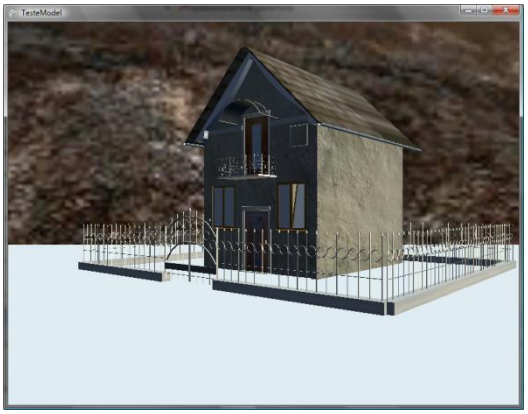


Figura 2 Modelo de uma casa, visão transversal

Estas imagens apresentam o modelo de uma casa em 3D, a primeira imagem representa a posição inicial da casa, já a segunda demonstra uma rotação no eixo do Y que para este motor representa o eixo vertical. Utilizando os cursores do teclado consegue-se alterar a posição da câmara em relação à casa, dando a ilusão de que é esta que se esta a mover. A próxima imagem mostra que se pode mover a câmara para qualquer posição, mas sempre apontada a casa.

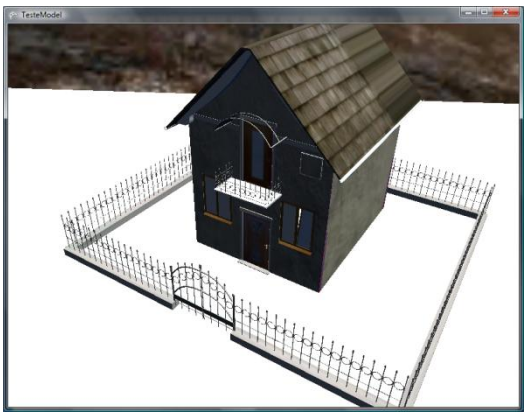


Figura 3 Modelo de uma casa, visão 3d

Estes tipos de simulações podem ser facilmente criadas com recurso aos motores de jogos, facilitando assim a demonstração de projectos. Uma empresa de construção civil poderia utilizar estas ferramentas para demonstrar os seus projectos dando uso assim a uma ferramenta que, apesar de não ter sido criado originalmente para este efeito, seria um óptimo apoio para as apresentações.

3.7. Programação de simulações com ferramentas de desenvolvimento de jogos

Ferramentas dedicadas à programação de jogos podem ser utilizadas para as simulações interactivas e os modelos de software são os mesmos que são utilizados para o desenvolvimento de sistemas comuns. Deste modo a programação de simulações interactivas é facilmente traduzida a partir de um jogo de computador.

3.7.1. XNA Game Studio

O exemplo apresentado no ponto 3.6 deste capítulo, foi desenvolvido no motor de jogos do XNA, mas os métodos utilizados não são necessariamente métodos dedicados à simulação, nem são métodos específicos do XNA. A casa é apenas um modelo 3D da mesma, criado em 3DStudio, mas a câmara que esta a ser utilizada para ver a casa foi criada a partir de um código de um jogo, onde a câmara mantinha-se sempre numa visão de 3ª pessoa. Percebe-se então que o XNA é o motor que mostra todos estes artefactos, mas os métodos usados não necessitam de ter sido exactamente criados para o efeito que se está a dar.

Nos próximos capítulos o XNA Game Studio será explicado em mais detalhe, fica só aqui a noção de que com um motor dedicado ao desenvolvimento de jogos, pode ser facilmente utilizado para a criação de simulações interactivas.

3.7.2. Projecto Scratch do MIT

Este projecto é um exemplo ilustrativo da importância desta área de desenvolvimento de software, principalmente na área de aprendizagem de lógica. Este software oferece uma perspectiva de construção de jogos e simulações à base de blocos de dados, ao estilo de Legos, assim permitindo que os programadores possam ter idades de 8 anos ou superiores, mas conseguirem criar os próprios jogos.

Figura 4 Blocos de Scratch

Esta figura apresenta os blocos de dados de teste e dos ciclos que são usados no Scratch, pode-se usar os "se" e "senão" tal como se usa numa linguagem de programação, apenas neste software a visualização é diferente, assim é fácil perceber a facilidade que existe para se usar esta linguagem no ensino a crianças. Com o uso do "drag and drop" a programação simplifica-se e oferece ao utilizador uma facilidade como nenhuma outra ferramenta de programação oferece.

Para se compreender melhor a programação por blocos pode-se usar um exemplo de um se tal como está na figura seguinte. Esta demonstra que caso uma variável "i" seja igual a 10 que o objecto se moverá 10 passos. Senão o objecto roda 15 graus no sentido dos relógio.

Figura 5 Bloco Se do Scratch

Este projecto é explicado com mais detalhe no capítulo 5, apresentando um pouco da programação de um pequeno jogo e da interacção entre o utilizador e o software.

4. Industria dos Jogos de Computador

4.1. Introdução

Quando se pensa em jogos de computador, somos logo invadidos pela ideia do lúdico. À partida os jogos de computador são produzidos para brincar, no entanto o processo de construção de um jogo revela-se em algo bem mais desafiador do que se possa imaginar.

Na construção de um jogo, lúdico ou não, tendo em conta o público-alvo a que se destina, para atingir um determinado objectivo, idealizado, estruturado, escrito e construído pela imaginação, ter-se-á de contar com um conjunto muito alargado de conhecimentos informáticos que servirão de base e suporte a toda a criação do jogo.

A história, os cenários, os enredos, a criação de níveis, os prémios, bonificações, punições, etc., são informações com que o programador deve trabalhar para a realização do jogo.

A programação de um jogo de computador é possivelmente uma das áreas mais complexas, em termos de algoritmos, no ramo da informática. Para se poder construir um jogo é necessário a compreensão e domínio total de várias matérias, que se inter-complementam num vasto processo de criação/produção. Durante todo o processo de criação de um jogo, são necessários cálculos matemáticos e físicos, conhecimentos de design gráfico, de bases de dados linguagens orientadas a objectos e de linguagens orientadas ao desenvolvimento de jogos.

Por exemplo: só um simples mudar de posição de um objecto, necessita de cálculos matemáticos que, quando não compreendidos, podem complicar ou atrasar o processo de criação do jogo.

Dado que todo este conhecimento por si só não é o suficiente para atingir o sucesso, teremos que adicionar ao conceito do jogo, o *gameplay*, a história e mesmo a interacção com o jogador, que são elementos essenciais para o produto final. Isto demonstra que um jogo é, necessariamente, mais complicado de desenvolver do que um programa informático. O processo de criação de um jogo de computador, necessita de uma vasta equipa de pessoas, com diferentes características, capazes de complementar

os seus conhecimentos focalizando-os no objectivo a que se propuseram para que o produto final constitua algo de inovador, coerente e com qualidade superior, no sentido de conquistar o mercado associado.

Sintetizando o jogo de computador não é apenas um brinquedo, mas sim um grande e complexo projecto de software, que foi desenvolvido por uma vasta equipa de profissionais. Por exemplo se todos os programas de software se preocupassem em tirar o melhor partido possível da máquina, ou a apresentar o melhor ambiente ao utilizador, como nos jogos, estes poderiam ser ainda mais eficientes. A evolução continua, o que permite compreender o vasto conjunto de aplicações empregues nos jogos a serem utilizadas no desenvolvimento de aplicações administrativas, técnicas e científicas. Com o desafio da construção de novos jogos seremos sempre confrontados com novas técnicas, criações e efeitos surpreendentes; os jogos não se fecham sobre si mesmo, os resultados têm e terão cada vez mais aplicações em todos os campos do saber.

4.2. História Recente da Programação de Jogos

O jogo foi sempre uma actividade de treino, teste e preparação para a defesa, para a caça, para além do aspecto lúdico desde o início da humanidade.

Foi em 1958 que os jogos passaram à era digital e entraram no mundo dos computadores, isto quando William A. Higinbotham desenvolveu o jogo TENNIS FOR TWO. [Nacke 2005]. Desde então a indústria de jogos de computadores tem evoluído, até, mesmo, ultrapassar a indústria da música e do vídeo. É actualmente uma das indústrias mais fortes no mundo das tecnologias.

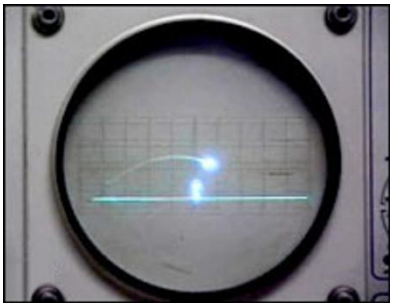


Figura 6 Tennis for two

A história continua e, em 1962, um aluno do Massachusetts Institute of Technology (MIT) desenvolveu o primeiro Space Wars para um dos computadores mais potentes da altura, o PDP-1. Este jogo criado com a intenção de testar as capacidades do computador, foi um sucesso junto da maioria dos alunos das faculdades americanas que deram por si a jogar o jogo em vez de ir para as mesas de pinball. Em 1971 foi distribuída a primeira máquina de jogos a moedas. A partir daqui surgiram companhias como a Atari e outras que conceberam consolas para ligar aos televisores.

Foi na década de 80 que começou a revolução das consolas com o surgir da Nintendo. Com as primeiras NES, o Japão entrou em força na guerra das consolas. Ainda nessa década surgiram outras consolas de renome, como sejam a Sega, com a Master System, e a Atari com o novo Atari 7800. Nesta altura a Nintendo vencia a concorrência ao conseguir vender 10 vezes mais consolas que os seus principais opositores. No final desta década apareceu a primeira grande consola portátil: a Nintendo Gameboy.

A partir daí as grandes consolas desenvolveram-se estando sempre a ser melhoradas ou acrescentadas. A NES, Master System, Megadrive, Super NES, NeoGeo, Saturn, PlayStation, Nintendo64, Dreamcast, Playstation 2, Xbox, Gamecube, Xbox 360, Wii e Playstation 3 são as principais consolas do mercado de acordo com a sua época.

Em simultâneo com a evolução das consolas, também o PC, com as características que já dispunha e com a entrada das placas gráficas, tipo 3dfx, NVIDIA e ATI, entrou na guerra dos jogos. Assim, e neste momento, as 4 principais plataformas de jogos são o PC, a Xbox 360, a Playstation 3 e a Nintendo Wii.

Os jogos evoluíram juntamente com a evolução dos computadores e das linguagens de programação. Uma linguagem que foi originalmente criada para desenvolver um sistema operativo (UNIX) foi mais tarde reutilizada para o desenvolvimento dos jogos e assim associada para sempre à sua criação. Esta linguagem é o C, que evoluiu mais tarde para o C++, e que passou a ser uma referência na programação dos jogos, continuando a ser uma das melhores ferramentas no desenvolvimento dos melhores jogos de sempre. Hoje em dia já existem várias ferramentas feitas especificamente para o desenvolvimento de jogos, mas podemos continuar a dizer que a industria evoluiu a partir do C.

Dentro do mundo dos jogos também se deve mencionar que existem vários tipos de jogos, que por sua vez apresentam maneiras diferentes de visualizar a realidade ou mesmo de a alterar. Estes podem ser visualizados pelo jogador na primeira, na terceira pessoa ou mesmo como uma entidade externa. Podem ser jogados de diferentes maneiras, tais como utilizando comandos, teclados ou mais recentemente por gestos. Facilitam ainda a comunicação entre diferentes comunidades do mundo, graças à

internet que é usada para partilhar aventuras que seriam impensáveis há uns anos atrás. Isto apenas vem confirmar que os jogos estão sempre em constante evolução, o que nos força a imaginar como será o mundo dos jogos daqui a dez anos.

Os jogos de computador inserem-se ainda num conjunto das mais variadas actividades em que se pode melhorar a atenção, aumentar a velocidade de reacção, testar a inteligência, a aprender, a apreender novos conhecimentos e até mesmo, mais recentemente, levar as pessoas a fazer exercício físico, dançar ou relaxar.

4.3. A Indústria dos Jogos de Computador

A indústria dos jogos de computadores, mais formalmente conhecida como indústria de entretenimento interactivo é o sector económico do desenvolvimento, marketing e venda dos jogos de computador. Formada a partir da necessidade de gerir todo o mundo dos jogos de computador no final do século XXI, incorpora dezenas de competências e dá trabalho a centenas de milhares de pessoas por todo o mundo.

Esta indústria foi dos principais impulsionadores dos mais recentes desenvolvimentos de hardware e software para os computadores pessoais, desde as placas gráficas, placas de som e até sistemas operativos do género do UNIX, que foi original criado para correr um jogo de viagens espaciais. Hoje é conhecido como um dos maiores sistemas operativos para os super computadores ou grandes servidores do mundo. [Dennis 1996].

Outros desenvolvimentos também surgiram, as placas de som foram criadas para oferecer a possibilidade de inserir som digital nos jogos e mais tarde musicas e ficheiros de áudio. As primeiras placas gráficas foram desenvolvidas para apenas aumentar o número de cores, agora já estão a ser desenvolvidas placas que para além de oferecerem aceleração 3D também trazem cálculos preparados para funções de física e processadores dedicados (GPU’s) para se poder obter o máximo número de objectos em jogo, melhorando assim os efeitos especiais.

Os jogos modernos estão entre o software mais exigente para as capacidades de um computador, pois necessitam de uma capacidade de processamento que rivaliza com os maiores programas de edição de gráficos ou vídeo. Isto leva a que um grande volume das vendas dos grandes computadores seja efectuado a jogadores, que desejam sempre o

melhor para jogar, o que leva a que os jogos, sejam um grande impulsionador para o desenvolvimento das capacidades dos computadores.

4.4. Desenvolvimento de jogos de computador

Para o desenvolvimento de um jogo torna-se necessário começar por referir que é preciso saber, antes de mais, o que é um jogo. Basicamente um jogo começa por ser uma história que necessita de ser contada.

Ao longo dos tempos a maneira de contar histórias evoluiu: inicialmente o contador de histórias, relatava eventos de carácter étnico ou imaginativo, de carácter fantástico, que entretinham essencialmente a população mais nova, que o quisesse ouvir, e tentava transmitir-lhes alguns conhecimentos das raízes culturais desse povo. Posteriormente assistiu-se ao aparecimento do narrador que, como que em voz off, antecipava os acontecimentos que os actores exemplificavam em palco, explicando ao público o rumo da história. Este já se baseava em documentos escritos que deram origem aos livros de contos. Os livros tornaram-se na nova maneira de contar histórias, não tanto contadas, mas mais vividas individualmente por cada leitor. Após esta era e com a evolução tecnológica da humanidade a maneira de contar histórias evolui outra vez, agora para os filmes, que transcreviam para a tela as imagens dos livros ou as projecções do imaginário do realizador ou a sua interpretação. Assim sendo sempre existiram histórias o que variou foi o suporte de transmissão ou comunicação dessa história que passou pela transmissão oral até ao suporte fílmico. Com o suporte digital e com a programação de computadores dá-se uma revolução num curto prazo de tempo (30 a 40 anos).

A história libertou-se do texto e do guião, passando a também depender do leitor, que como jogador, pode interferir na estrutura da história, no seu desenvolvimento, até nos personagens podendo fazer com que a história/jogo possa ser vivida interactivamente, sem ter um resultado pré-definido.

Os livros são baseados numa narração, que conduz o leitor a partir de uma ideia ao longo da história sem permitir a sua intervenção ou a possibilidade de alterar alguma coisa. Todo o ambiente é descrito normalmente em detalhe cabendo ao leitor o trabalho de o imaginar. Todos os argumentos são construídos segundo uma linha que é definida pelo pensamento do autor.

Os filmes, construídos segundo a interpretação do realizador, são em quase tudo idênticos aos livros, agora conduzidos pelo realizador, com algumas excepções; o argumento, a história, o ambiente são transmitidos por uma representação visual, impedindo assim o leitor de imaginar o seu conteúdo, uma vez que já não tem de intervir com a sua imaginação mas apenas de percepcionar o que lhe está a ser comunicado visual e sonoramente, facilitando a sua compreensão mas perdendo por vezes os detalhes que fazem parte da história. Neste caso o espectador não necessita de ter imaginação, segue apenas o que lhe está a ser apresentado.

Ao contrário dos suportes anteriores, livros e filmes, onde o leitor/espectador é conduzido passivamente ao longo da história, actualmente, num jogo ele pode realmente alterar o desenrolar dos acontecimentos, é claro que o que o jogador pode fazer está limitado pela programação do jogo, mas mesmo assim, o jogo oferece uma característica única. Os jogos são os **livros** + os **filmes** + **interactividade**. A interactividade é a possibilidade do jogador alterar o seguimento do jogo, isto é da história.

Os jogos podem então ser avaliados pelo nível de interactividade; quanto mais interactivos melhor a experiência, logo, melhor será o jogo. *"Se não existir interactividade não é um jogo mas sim um conto."*

A interactividade é a característica intrínseca dos jogos de computador que os diferencia de todos os outros suportes já abordados. Por exemplo, um jogador pode ter como objectivo ver o final da história, sem ter de percorrer vários trajectos que lhe são apresentados, dando-lhe assim a possibilidade de escolher qual dos percursos que quer. Isto permite ao intérprete/jogador seleccionar o caminho que deseja, criando a sua própria versão da história. Num jogo ideal nada está predefinido, tudo depende da opção de quem joga, obviamente isto ainda não se vê nos jogos de hoje em dia porque necessitaria de uma capacidade infinita de possibilidades programadas antecipadamente e a capacidade de processamento de parte do jogo.

O processo de programação que se segue pode ser caracterizado de acordo com a dificuldade de construção e que pode ir do mais ou menos complexo ao de extrema dificuldade, porque para criar e concretizar um jogo há um processo longo, rigoroso e difícil. De acordo com o que se pretende um jogo de qualidade pode demorar anos a ser

desenvolvido e, mesmo quando acabado, pode conter inúmeros problemas que deverão ser resolvidos posteriormente por upgrades/updates.

Durante todo este processo, e durante todas as fases é necessário que a ideia e a concepção do jogo se mantenham iguais para todos os intervenientes no projecto, de modo a que todos estejam a trabalhar para um mesmo objectivo. Para isso é necessário criar, logo no início, um documento que registará e acompanhará todo o projecto até ao seu termo. Este documento, ou guião, tem usualmente no meio informático a designação de Game Design Document. A criação deste documento está sujeita a uma estrutura em que são definidos os mais diferentes passos. Primeiro terá de definir a ideia daquilo que se pretende que o jogo seja. Do tipo de jogo, ao tema, público-alvo, conteúdo, desenvolvimento progressivo, e mesmo que tenha um espaço para que algum item possa ser retirado ou acrescentado ao projecto final.

Assim, tal como foi dito, são necessárias várias etapas que permitem estruturar e facilitar o desenvolvimento de um projecto/jogo.

5. Engenharia de Software para a Construção de Jogos

5.1. Introdução

Este capítulo demonstrará que a engenharia de software na programação de jogos de computador é semelhante, quando não igual, à de um sistema normal de software. No capítulo de Programação de jogos já se falou de vários passos necessários que são semelhantes, tipo o Documento do Processo de Desenvolvimento do Jogo (Game Design Document - GDD) que se assemelha ao documento de requisitos. Este capítulo irá demonstrar, todas as fases de criação de um jogo, mas de modo a criar um paralelismo entre o desenvolvimento de um software e de um jogo de computador.

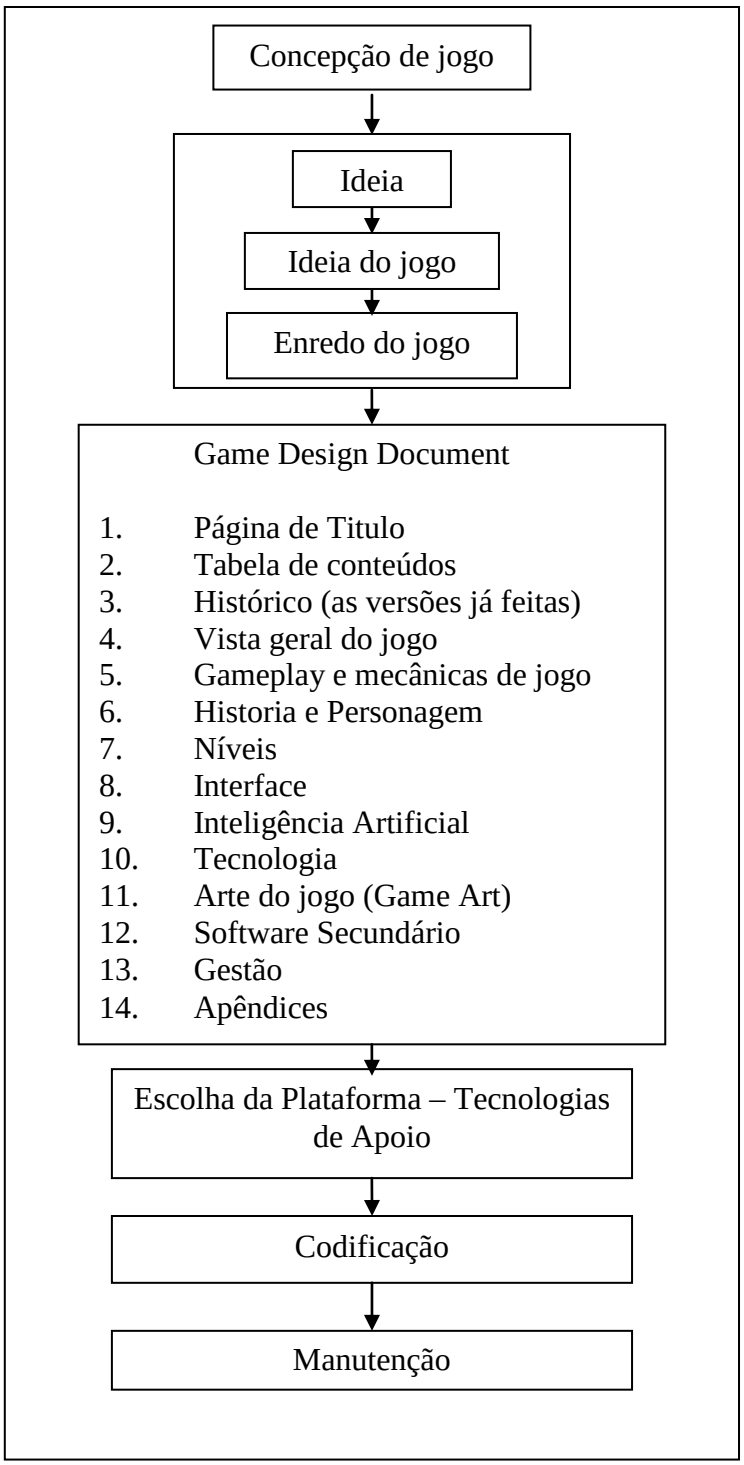
O intuito é provar que com as mesmas técnicas, linguagens de programação e conhecimentos, na área da informática, se pode criar um jogo de computador.

5.2. Processo de criação de um jogo

O Processo de criação de um jogo de computador e de um produto de software são muito semelhantes. Metodologias tipo Ágil e RAD também são usadas para a criação de jogos. Estas metodologias não são necessariamente específicas para o desenvolvimento de jogos de computador, mas sendo o jogo um software este terá o mesmo ciclo de vida de um software normal assim podem e devem ser usados os mesmos modelos.

Em Engenharia de Software são necessários vários passos antes de se entrar na parte de codificação. Em primeiro lugar têm de se preparar os recursos de software, e prosseguir pela fase de design. No caso de criação de jogos, estes passos são introduzidos na criação do Documento do Processo de Desenvolvimento do Jogo (Game Design Document - GDD). Este documento contém todos os detalhes necessários à criação do jogo. Inicia-se o projecto criando a ideia de jogo que, juntamente com a análise de requisitos e design de software, irá criar o Game Design Document (GDD). Com este documento será possível passar à fase de codificação e começar a desenvolver o jogo. A tabela seguinte apresenta o modelo como foi idealizado, será fácil perceber que o modelo não diverge muito do modelo que é apresentado para Engenharia de Software, mas com umas determinadas alterações de modo a ser orientado para a programação de jogos.

Engenharia de Software orientada à programação de jogos e simulações interactivas



Esquema 10 Modelo de Engenharia de Software orientada à programação de jogos / simulações interactivas

5.3. Design de Jogos

Nesta fase pretende-se estabelecer o objectivo do jogo. Para o desenvolvimento de jogos de computador, antes de se escolher metodologias e ferramentas, é necessário conceber o jogo no seu todo. O Design é o processo e suporte para criar o jogo, de o imaginar, definir o seu funcionamento, descrever os elementos que o constituem e transmitir estes elementos para a equipa de trabalho.

A concepção pode começar por ser apenas uma simples ideia que quando desenvolvida e aperfeiçoada se concretiza no jogo. Um exemplo disso e que tem ganho muitas atenções especialmente pela sua jogabilidade é o "World of Goo". Este jogo que apenas usa um simples motor 2D consegue ser óptimo por causa da sua jogabilidade. O conceito do jogo é simples: trata-se de levar uma série de bolas de petróleo (Goo) para um cano. A ideia não é nada de inovadora, mas o facto de os criadores usarem as leis da física para darem movimento às bolas em todo o jogo, ajuda a criar um ambiente cativante o que motiva o utilizador a seguir em frente.

5.3.1. Ideia de Jogo

Ideias simples constituem grandes jogos, um exemplo muito conhecido é o de dois canalizadores que querem salvar uma princesa de um terrível dragão.

As ideias iniciais não precisam de ter sentido, não necessitam de grandes descrições, podem tomar como ponto de partida elementos simples a serem desenvolvidos pela imaginação de quem a está a formular. Todos os jogos começam assim, nada mais é necessário para começar um jogo, neste ponto a única coisa que é preciso é a descrição da personagem. Ou seja, quem é a personagem e o que ele faz, assim já se está a dar uma envolvência ao jogo, pois se soubermos quem é a personagem já se consegue imaginar a continuação.

Uma vez criada a ideia inicial tem de ser desenvolvida a ideia de jogo. Esta ideia que não é mais que a estrutura do funcionamento do jogo, não deve conter mais de 8 a 10 linhas, que apenas descrevem os desafios que o jogador irá ter de ultrapassar e o que fazer para os ultrapassar. Neste ponto, tem de se descrever todos os obstáculos que o jogador terá e como os conseguir superar, por exemplo: o jogador poderá ter a

possibilidade de saltar por cima dos inimigos e se lhes cair em cima eles morrem, caso contrário será o jogador a perder vida.

5.3.2. Concepção de jogo

A maior dificuldade no design não é apenas ter a ideia mas sim o desenvolver do funcionamento de um jogo desafiador e divertido. Pois se este não desafiar o jogador ao mesmo tempo que o diverte não merece ser jogado e assim a ideia perde o seu valor. É neste tempo de concepção do jogo que se torna necessário ultrapassar esta dificuldade.

Ao conceber o jogo o designer terá de ter em mente que este deve entreter e cativar as outras pessoas. Isto é a interactividade do jogo, quanto melhor for mais o jogo entreterá. O conceito deverá ter 4 elementos principais. A Narrativa, o Papel principal (Role), a descrição / Setting (quem somos) e o Gameplay (o que se pode fazer).

Narrativa – Caso seja necessária uma história, esta será introduzida neste ponto (mas existem jogos que não necessitam de história (como o Tetris)). O jogo necessitará de um história que será percorrida pelo jogador sendo para tal criados mundos no seu percurso que uma vez desenvolvidos farão parte da história.

Papel e Objectivos – Neste ponto do projecto/jogo deverá ser descrito o interveniente na história ou personagem, quais as suas características e objectivos que intuitivamente podem sugerir o que ele pode fazer, movimentos, limitações, apetências e potenciação. O papel do jogador é a peça chave para o gameplay do jogo.

Cenário – Caracteriza o espaço onde se desenrola o jogo. Pode ser mais do que um e se for tridimensional deve permitir o seu percurso em vários sentidos. É o modelo de interacção, a maneira como o jogador interage com esse espaço/mundo (se um avatar, se é omnipresente, etc.). É a perspectiva de como o jogador vê o jogo (1ª ou 3ª pessoa). Finalmente o mundo do jogo.

Gameplay – A diversão, o jogo deve ser prazenteiro e divertido. Este objectivo atinge-se formalmente pela qualidade, ou seja pela quantidade de opções que o jogo oferece. O jogador interage com o jogo escolhendo opções (correr, saltar, andar, etc.) ultrapassando obstáculos ou não tendo um resultado imediato (ganha ou perde). Um bom Gameplay significa um jogo que é repetível, permitindo voltar atrás em qualquer

situação, o que faz com que o jogo se torne viciante. Esta é uma das características necessárias para que um jogo possa ser um sucesso.

Na realidade a maioria dos jogos copiam temas e situações que já existem na nossa sociedade quotidiana. Facilita-se assim a criação do jogo e oferece-se à equipa de trabalho a possibilidade de no seu desenvolvimento se focar mais no jogo em vez de perder tanto tempo na história.

5.3.3. Temas do Jogo

Os temas dos jogos para computador podem ser resumidos em três tipos de lazer: o Social, Cultural e Desportivo sendo que dentro destes três tipos podem-se desenvolver imensas concepções para jogos, tipo:

Social

Festas e eventos sociais, amizades e novos conhecimentos, bem como a criação virtual de uma vida quotidiana, podem ser usados como temas de jogos, existindo já jogos que utilizam este tipo de ocupação lúdica, como por exemplo o SIMS ou o Second Life. Estes jogos conseguem aproximar o utilizador destes eventos e da sua simulação virtual. Para além do seu claro aspecto lúdico, este tipo de jogos pode ter a sua mais-valia na apreciação e avaliação de comportamentos na área da psicologia e da sociologia. Na simulação de situações e no seu estudo correspondente. Com a programação e as técnicas desenvolvidas para jogos podem proceder-se a estudos de circulação viária dentro de cidades, em feiras e até dentro de edifícios.

Cultural

Livros, Filmes e Teatro oferecem, por si só, praticamente tudo o que um jogo necessita para o seu desenvolvimento. Já foram construídos vários jogos que se baseiam na história de um bom livro ou filme, como por exemplo todos os jogos das sagas Star Wars ou Senhor dos Anéis. Neste grupo de jogos são muito conhecidos os que foram desenvolvidos para acompanharem o desenvolvimento de crianças nos vários grupos etários para as diferentes disciplinas escolares. São também conhecidos os jogos que põem à prova os nossos conhecimentos artísticos de música, na pintura, na visita virtual a museus, palácios e até a cidades. Trata-se de uma área por explorar e com uma variedade quase infinita de jogos.

Desportivo

Os jogos baseados em temas de desporto são possivelmente os mais comuns, sendo que podem ser desenvolvidos com a possibilidade de jogar individualmente ou em equipa. Os jogos de futebol, de basketball, de ténis, etc., que necessitam de controlos de jogo em equipa, são os mais conhecidos. É neste aspecto que este grupo se distingue dos restantes uma vez que exige a participação de mais do que um jogador. Pode-se perceber facilmente a potenciação deste tipo de programas/jogo com aplicações, por exemplo, do tipo de defesa (situações de guerra, de terrorismo, policiais, manifestações, etc.) que tanto podem ser criadas para uso lúdico como uso científico.

5.4. Concepção do Jogo

Antes de criar o GDD é necessário desenvolver a concepção de jogo, na realidade ter a imaginação para construir uma história inovadora, aliciante e diferente de modo a cativar o seu desenvolvimento. Um exemplo de será apresentado nos próximos pontos de modo a ter um documento sensivelmente completo e de fácil explicação. Nem todas as secções do plano do GDD apresentado poderão ser necessárias, pois depende do tipo de jogo que se está a desenvolver.

5.4.1. Ideia

A ideia inicial será algo muito simples e a que será usada é, "*O mundo está a ser atacado por algo misterioso, serão extraterrestres, serão monstros não se sabe, mas com a nave iremos repeli-los*". A partir daqui já se possui um ponto de arranque para o jogo, temos o jogador, que será a nave, algum contexto para os inimigos e para a história, agora ter-se-á de adaptar esta ideia à ideia do jogo.

5.4.2. Ideia de Jogo

Seguindo a ideia criada, ter-se-á então a nossa nave que irá defender o mundo de um inimigo, estes serão representados por naves de extraterrestres que querem invadir o nosso mundo. O cenário de fundo onde se desenvolverá a acção é representado pelo mundo que será observado de uma perspectiva lateral, ou seja ver-se-á o jogo a decorrer num sentido horizontal e as naves poderão andar em todos os sentidos horizontais e verticais. No caso dos inimigos, movimentar-se-ão no sentido oposto ao do jogador com

velocidades diferentes de modo a tentar embater contra o mesmo, eliminando-o. O jogador terá a possibilidade de se afastar dos seus inimigos e também disparar contra eles. O jogo terá ainda um "chefe" no final de cada nível, que terá um tamanho superior ao dos inimigos normais e poderes igualmente superiores. Para conseguir ultrapassar cada nível o jogador terá de derrotar este "chefe". Durante o jogo, irão surgir também alguns upgrades para o jogador, tipo melhores tiros, ou mais vidas, estes upgrades deverão aparecer a flutuar no ecrã com movimentos aleatórios.

Para uma ideia de jogo, esta será o suficiente, pois agora já sabemos como é o contexto do jogo quais os possíveis movimentos do jogador e do inimigo, como decorrerão os movimentos e também alguns acrescentos que o jogo terá. Certamente que esta não é uma ideia de jogo inovadora, mas quanto mais complicada fosse a ideia mais complicado seria explicar o processo.

5.4.3. Enredo de Jogo

Esta ideia de jogo já demonstra parte da narrativa, pelo menos uma apresentação dela, mas pode-se criar uma história para o jogo. Hoje em dia fala-se muito na chegada de um planeta novo ao sistema solar, assim pode-se criar a história que nesse novo planeta existe uma civilização tecnologicamente mais desenvolvida e que, já da última vez que este planeta passou pelo sistema solar, atacaram o nosso planeta, na altura quase tudo foi destruído mas desta vez estaremos preparados com um esquadrão de naves elite para defender o nosso planeta. Os extraterrestres, desta vez, não terão a vida facilitada. Este esquadrão chama-se SAD, "Seek and Destroy". Esta força de elite possui 5 naves de alto desempenho para defesa do nosso planeta e serão requisitados para atacarem os novos inimigos defendendo o planeta.

Papel e objectivos

O papel principal deste jogo será a nave do jogador, que tal com já foi descrito poderá usar uma das 5 naves do esquadrão; estas terão movimentos idênticos mas apresentam algumas características diferentes umas das outras. As naves poderão movimentar-se em todos os sentidos horizontais e verticais sobre o cenário do mundo e terão a possibilidade de disparar contra os seus inimigos. Mas cada nave possui diferentes características. A primeira nave terá uma velocidade superior em relação às outras, a segunda terá maior resistência aos ataques, e a terceira terá mais poder de fogo.

Quanto às duas últimas, uma será maior mas também será mais resistente e a ultima será mais pequena e rápida mas com menor resistência em relação a todas as outras.

Cenário

O cenário para o jogo já foi sucintamente descrito. Já se sabe que o personagem será uma nave, logo o avatar será esta nave, que será vista de cima numa perspectiva de 3ª pessoa. O mundo será visto de uma perspectiva lateral, ou seja veremos o mundo a passar por traz da nave. O movimento do mundo efectuar-se-á da direita para a esquerda e a nave terá de voar na direcção oposta para poder chegar ao seu destino.

Gameplay

No caso do jogo que está a ser desenvolvido, o *gameplay* é simples, pois a nave pouco mais pode fazer do que voar e disparar. Serão então os vários upgrades que criarão a qualidade do jogo. Os níveis também serão primariamente diferentes, assim mantendo a atenção do jogador nos novos mapas e inimigos.

5.5. Documento do Processo de Desenvolvimento do Jogo

O Design do jogo é o processo de registar o jogo imaginado. Após todos os passos anteriores será redigido todo o documento de design do jogo que irá acompanhar o trabalho de todos os programadores, designers, artistas, etc., que compõem a equipa e que constituirá o suporte base e estrutural do desenvolvimento de todo o jogo.

"*O Game Design Document (GDD) é o mapa do qual o jogo será criado. Como tal, todos os detalhes necessários para a construção do jogo tem de estar mencionados no documento. Se não estiver no documento, então provavelmente não estará no jogo*". [Baldwin, 2005]

O GDD é o guião e o mapa sobre o qual o jogo será construído. Como tal, deverá conter toda a informação que se refere ao jogo desde a ideia, os participantes, os cenários, e todos os detalhes necessários para a sua construção. O documento mais parecido com o GDD é o guião de um filme de animação, mas que no caso de um jogo tem de especificar de forma clara e convincente a interacção com o jogador ou jogadores, deve indicar claramente o objectivo final do jogo, propósitos e opções. É como já referi o documento base de trabalho de toda a equipa de desenvolvimento que

disporá de documentos parcelares técnicos, fichas artísticas e outros que deverão ser mencionados no GDD.

Obviamente este documento é diferente de jogo para jogo, variando consoante a complexidade de cada um, da maior ou menor exigência de opções, objectivos, alternativas, etc. Podem-se imaginar os documentos criados para os jogos tipo PacMan, Super Mario, ou Quake. Todos estes jogos necessitam de GDD detalhados sendo que estes serão completamente diferentes nas suas características, modelos e até formatação. O documento define a maneira como o jogo funciona, descrevendo os elementos que compõem o jogo e transmite essa informação, de forma clara, para todos os membros da equipa que o está a desenvolver.

Desta forma combinam-se todas as formas necessárias para o desenvolvimento do jogo: a ideia, a arte, a ciência e a habilidade no mesmo documento.

Segundo o seu criador o "Game Design Document" deve conter os seguintes pontos principais: Mecanismos Principais (Core Mechanics), Narrativa/História e Interactividade. Mas é óbvio que não é constituído apenas por estes três pontos. É a partir deles que é desenvolvido o GDD.

5.5.1. Mecanismos Principais

São o cerne do jogo. São as regras que o regulam e definem a maneira como o jogo funciona. Por exemplo as regras de um jogo de xadrez, não são necessárias para mexer as peças mas são necessárias para jogar o jogo.

5.5.2. Narrativa e História do jogo

Todos os jogos contam uma história, por muito simples e dissimulada que esta seja, existe sempre. A narrativa é a parte da história que o autor conta ao jogador, a parte não interactiva do jogo. O jogador pode alterar o desenrolar da história mas nunca pode alterar a história em si, a interactividade do jogador acaba quando começa a narrativa do autor.

5.5.3. Interactividade do jogo

A interactividade representa a forma como o jogador interage com o jogo. O que o jogador faz, ouve, vê e realiza no mundo virtual. O Game Designer define todas as interacções possíveis entre o jogador e os objectos do jogo. Quanto maior for a interacção melhor será a participação do jogador no jogo. Um bom jogo é avaliado pela vontade que os jogadores demonstram em continuar a jogar mesmo depois de o terem terminado. Um jogo com boa interactividade é um bom passo para atingir essa finalidade.

Como foi descrito um GDD é necessariamente diferente para todos os jogos o que torna complicado desenvolver uma estrutura que possa ser seguida independentemente do tipo de jogo. Mas já existem alguns que se aproximam muito dessa realidade.

Dentro dos três pontos já descritos pode-se planificar melhor o jogo, assim dando origem ao seguinte índice:

1. Página de Titulo
2. Tabela de conteúdos
3. Histórico (as versões já feitas)
4. Vista geral do jogo
5. Gameplay e mecânicas de jogo
6. Historia e Personagem
7. Níveis
8. Interface
9. Inteligência Artificial
10. Tecnologia
11. Arte do jogo (Game Art)
12. Software Secundário
13. Gestão
14. Apêndices

Tabela 1 Índice do Game Design Document

Este índice pode ser usado no GDD de qualquer jogo, pois mesmo que alguns dos parâmetros não seja necessário para o jogo, pode ser simplesmente omisso.

Seguidamente ir-se-á aprofundar melhor cada um destes parâmetros, demonstrando assim exactamente como um GDD deve ser utilizado, mesmo que algumas partes não sejam totalmente necessárias.

1. Página de Titulo

Nesta página será necessário inserir, para além do nome do jogo, a informação de direitos de autor, o número da versão do jogo, o autor e a data.

2. Tabela de Conteúdos

A tabela de conteúdos, tal como o nome diz, é um índice que contém todas as subsecções do documento, é necessário ter a certeza que todas as secções estão representadas nesta tabela.

3. Histórico

Esta é uma lista que descreverá todas as grandes mudanças das versões do jogo sempre que este é modificado com o objectivo de o melhorar. As alterações têm de ser introduzidas nesta secção.

4. Vista Geral do Jogo

A vista geral do jogo será dividida em mais uma série de subsecções, a concepção do jogo, que envolve o conceito e ideia inicial que levou à criação do mesmo, as características iniciais do jogo, tipo de jogo (RPG, FPS entre outros), o publico alvo, ou seja, a população a quem o jogo se destina e qual a idade mínima legal para o utilizar, o sumario do game flow (como o jogador se movimenta no jogo), look and feel, que descreve a visão básica do jogo e como este se deve comportar. Por último temos o Project scope, que é um sumário de todos os principais locais e ambientes do jogo.

Esta última subsecção envolve o número de localizações, níveis, npc's (personagens não humanas do jogo), número de armas disponíveis, entre outros.

5. *Gameplay* e mecânicas de jogo

Nesta secção é descrito o plano do jogo dividido nas seguintes secções, gameplay, mecânicas/regras do jogo, screen flow (descrições dos ecrãs/cenários e como se relacionam), opções do jogo, save e load e por ultimo ajudas.

A secção de gameplay é constituída pela progressão do jogo, bem como a parte que envolve a estrutura das missões e dos puzzles. No gameplay é também inserido os objectivos do jogo e o game flow, sendo este último o modo como o jogador tem a percepção do decorrer do jogo em si.

Na secção de mecânica e regras do jogo, são definidas quais as regras do jogo, tanto as implícitas como as explícitas. Estas regras envolvem todo o mundo que se encontra a ser simulado no jogo bem como a forma como os objectos interagem. Tudo isto implica conhecimentos de várias áreas tais como, física, regras de movimento, objectos, acções, combates, economia entre outros.

6. Historia e Personagem

Nesta secção irão ser descritas todas as histórias e personagens do jogo. Secções como historial das personagens, progressão no jogo, Cut Scenes, serão representadas. Também será apresentado o mundo do jogo, características físicas e descrições gerais do ambiente.

A história e narrativa envolvem, todo o historial do jogo, elementos de enredo, progressão do jogo, licenças necessárias e cut scenes. O mundo do jogo é constituído pelas imagens e ambiente do mesmo e pelas áreas ou níveis. Estas devem integrar a descrição geral, características físicas, níveis e conexões a outras áreas. Isto deve ser repetido para cada área ou nível do jogo.

As personagens devem ser descritas de acordo com o seu historial, personalidade, aspecto, habilidades especiais, relevância para a história e para com as outras personagens. Isto deve ser realizado para casa personagem integrante do jogo.

Esta secção está dividida em três partes de modo a separar a história em geral do mundo do jogo, as áreas do jogo e as personagens. Deste modo poder-se-á trabalhar em separado cada uma das partes.

7. Níveis

Aqui serão explicados ao mínimo detalhe cada um dos níveis do jogo. O nível não é necessariamente uma área do jogo, daí estar separado da secção anterior. Uma área pode conter vários níveis, ou os níveis podem ser contabilizados de acordo com a

evolução do jogador e não da zona onde se encontra. Por estas razões os níveis têm de ser extremamente bem explicados.

Cada nível necessita então de ser descrito em detalhe. Para a concretização dessa finalidade podem-se usar os seguintes campos, sinopse, material introdutório (cut scenes), objectivos, descrição física, mapa, caminho critico, encontros, descrição do nível pormenorizadamente e material de fecho.

Desde que se sigam estes apontadores, facilmente se fará uma descrição completa de qualquer nível para um jogo de computador

8. Interface

A interface é outra das secções que necessita de ser bastante detalhada, pois será a partir desta que o jogador irá navegar pelo jogo, independentemente do tipo de jogo a que se destina.

Na interface deverá vir descrito o sistema virtual e de controlo do jogo, áudio e musica, efeitos sonoros e sistemas de ajuda.

Estes seis são facilmente explicados, o sistema visual engloba os menus, o sistema de renderização, as posições das câmaras e os modelos de iluminação. O sistema de controlo apresenta o modo como o jogador controla o jogo, que tipo de comando. Áudio, música, efeitos sonoros e sistema de ajudas, finalizam o interface, que se for bem executado poderá ser superior a muitos outros interfaces.

9. Inteligência artificial

A maior parte dos jogos, hoje em dia, necessita de inteligência artificial, assim torna-se necessário prever todas as jogadas que podem ser feitas pelos personagens do jogo e programá-las de modo a comportarem-se de acordo umas com as outras. Por exemplo no jogo de xadrez é preciso saber o modo como todas as peças se movem para poder criar estratégias e introduzi-las na inteligência artificial.

Esta secção engloba os seguintes pontos, IA do oponente, personagens que não entram na acção de combate (NPC’s), personagens amigáveis e IA de suporte. Este engloba, por exemplo, colisões e detecção de caminhos.

10. Tecnologia

Esta secção conterá o hardware alvo e todos os restantes pré-requisitos para poder usar este hardware. Por exemplo não se poderá usar XNA para uma playstation. Assim com a escolha da plataforma fica-se cingido ao software, ao motor de jogo e as linguagens de scripting.

11. Arte do jogo

Nesta secção irão ser demonstrados todos os desenhos orientadores do jogo, tipo desenhos das personagens, ambientes, guias de estilos, equipamento, cut scenes e outros objectos do jogo.

12. Software secundário

Este software secundário, poderá ser o software de instalação, editor de mapas ou aplicações de update.

13. Gestão

Na gestão é necessário conter, um calendário detalhado, recursos, análises de riscos e planos de testes.

14. Apêndices

Nesta secção poderá ser introduzido tudo o que se possa ter negligenciado nas secções anteriores.

5.6. Tecnologias de apoio ao desenvolvimento

Com o GDD terminado, o próximo passo é a escolha das linguagens de programação que serão usadas para criar o jogo, neste caso não basta, apenas, escolher a linguagem que se gosta mais, primeiro tem de se saber para que plataforma se esta a desenvolver. Isto é necessário porque dependendo da plataforma também a linguagem muda.

As tecnologias que existem para a criação de jogos são imensas, mas podem ser separadas por dificuldade, tipo de programação, requisitos, finalidade e necessidades de cada uma. A seguir indicar-se-ão quatro tecnologias que apesar de serem orientadas

para o desenvolvimento de jogos, divergem na sua complexidade e no tipo de jogo que se deseja desenvolver, e apresentar-se-á a tecnologia escolhida para a parte de codificação deste trabalho. Existem mais tecnologias do que estas de que ir-se-á falar, mas a título de exemplo estas chegam para demonstrar a disparidade que existe entre os vários tipos.

Se a ideia é criar um jogo de duas dimensões (2D) e não possuir muito conhecimento de criação de jogos ou mesmo de linguagens de programação, poder-se-ão usar editores de jogos do género do GameMaker [pagina oficial GameMaker] ou do Scratch [pagina oficial Scratch]. O primeiro editor de jogos oferece uma grande variedade de ferramentas especificamente para o desenvolvimento de jogos 2D. Usando esta ferramenta um utilizador não experiente pode facilmente criar o próprio jogo de computador.

É claro que esta ferramenta tem as suas limitações, como qualquer outra, mas para se criar um jogo simples, para entrar no mundo de criação de jogos, ou fundamentar alguma ideia, este software será suficiente.

Por exemplo um programador de jogos mais experiente pode usar esta ferramenta para testar o gameplay de um jogo, ou seja a sua jogabilidade.

No caso do Scratch também é orientado a jogos 2D tal como já foi dito, mas este motor, criado no MIT [pagina oficial Scratch], leva a ideia de facilitar ainda mais longe. A ferramenta oferece uma serie de blocos que representam as funcionalidades básicas da programação e ao estilo de uma montagem de Legos, podem-se criar jogos. Devido a esta funcionalidade, virtualmente qualquer pessoa consegue criar o seu jogo, basta para isso investigar um pouco de como a ferramenta funciona.

Esta ferramenta foi escolhida para uma demonstração, devido à simplicidade que envolve. Será realizada uma comparação com o XNA Game Studio e será evidente a facilidade de desenvolvimento do Scratch.

É obvio que um jogo de última geração não depende de ferramentas tão básicas e sim de motores de jogo muito mais avançados, do género do Unreal Engine[pagina oficial Unreal Engine], este motor é usado para uma grande percentagem de jogos de

computador, visual e tecnologicamente avançados, jogos tipo Unreal Tournament, Gears of War, Medal of Honor e outros.

Este motor oferece uma grande variedade de ferramentas para a criação de cada segmento do jogo. Estas ferramentas oferecem a capacidade de renderização de objectos, animação, áudio, física, gameplay, scripting (sistema para a criação de sequências complexas de eventos), vídeos, efeitos de partículas, ferramentas de edição e uma linguagem de programação integrada (totalmente orientada ao objecto do género de C#).

Comparando estas tecnologias, apercebe-se da disparidade entre elas. O Gamemaker e é um editor de regras onde uma linguagem de programação nem sequer é considerada, o Scratch é uma linguagem ainda pouco desenvolvida e mais orientada ao ensino de crianças e o Unreal Engine contem um conjunto de ferramentas que necessitam um grande conjunto de conhecimentos para poder viabilizar o uso do motor mas quando usadas podem criar jogos de muito alto nivel.

Não falando sequer de custos, estas tecnologias contêm cada uma, um grande defeito. O Gamemaker e o Scratch são demasiado simples para o desenvolvimento de um jogo para o mercado, e o Unreal Engine é demasiado complexo, isto para o caso de um programador que queira iniciar-se no processo de criação de jogos.

Para simplificar o processo de criação de jogos, mas sem perder capacidades necessárias para um jogo que rivalize com os melhores do mercado, foi criado o XNA Game Studio [pagina oficial do XNA Game Studio]. Esta ferramenta criada pela Microsoft é uma biblioteca de programação baseada em C# que também contem uma Framework própria para o desenvolvimento de jogos. O que o XNA oferece com a Framework é um motor de jogos já funcional e totalmente preparado para que o criador, utilizando apenas as bibliotecas já existentes e algum conhecimento próprio, possa criar o seu jogo.

Por exemplo, o programador não tem de se preocupar em criar métodos específicos para ligar, testar e usar a placa gráfica, esta já vem especificada pela Framework, o programador apenas terá de a chamar e utilizar os métodos que a Framework oferece. O mesmo se passa com os cálculos matemáticos e físicos porque vêm inseridas na Framework bibliotecas que já especificam estes dados. Uma

característica que o XNA tem e que fez com que a ferramenta tenha evoluído muito depressa são as comunidades de desenvolvimento que possui, as outras tecnologias também dispõem destas comunidades, mas no caso do XNA estas são muito maiores. A razão para isto é o facto de a ferramenta ser livre.

Características / Tecnologias	Custos	Nível Dificuldade	Capacidade	Comunidades
GameMaker Pro	Razoáveis	Fácil	Baixa	Sim
Unreal Engine	Caro	Complexo	Alta	Sim
XNA GameStudio	Livre	Médio	Média/Alta	Sim
Scratch	Livre	Fácil	Baixa	Sim

Tabela 2 Comparação entre Tecnologias

5.7. Plataforma XNA

O XNA Game Studio é considerado um Kit dedicado à programação de jogos para quem deseja aprender a criar jogos de computador. Kit porque à partida a plataforma já oferece uma panóplia de ferramentas dedicadas a esse fim. O XNA traz incluído o IDE de desenvolvimento baseado no Visual Studio, mas apenas com a possibilidade de utilizar a linguagem de programação C#. A Framework, que é o conjunto de classes necessárias para a execução de um jogo XNA funcionando quer em .Net Framework para Windows ou .NET Compact Framework para XBOX e ZUNE. Traz ainda o XNA Content Pipeline, que é a componente de gestão de conteúdos do XNA, e para acabar o XACT, ferramenta de áudio para organizar todos os arquivos de som e os incorporar no jogo. Uma última característica é que o XNA Game Studio é gratuito o que permite um vasto conjunto de possibilidades para experimentar sem despesa inicial.

5.8. Codificação

Um dos primeiros conceitos que se tem de aceitar sobre a programação de jogos é que um jogo na realidade é um ciclo infinito de eventos, ou seja à semelhança com o que acontece com um filme o jogo é uma sequência de imagens que estão a ser tratadas à medida que o tempo passa. Tratadas porque para além de estarem a ser desenhadas no ecrã estão também constantemente a ser modificadas, mas com a diferença em relação o filme em que a interactividade com o jogador irá realizar a ideia de jogo.

5.8.1. Inicializar o Projecto de XNA

Ao iniciar um novo projecto em XNA, este cria desde logo o ciclo para o jogo e não só. Cria também todo o motor do jogo, basta correr o programa sem qualquer modificação que o motor já funciona. Apesar de que sem nada apenas irá aparece um ecrã azul, por traz desse ecrã já o ciclo está a correr e todas as classes do XNA estão prontas a ser usadas.

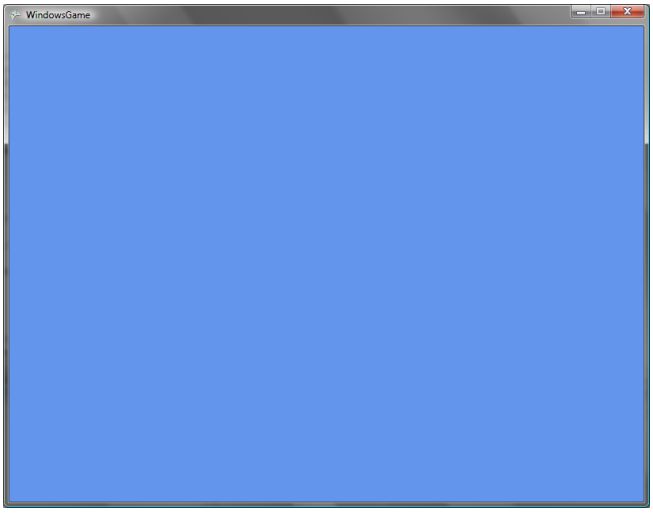


Figura 7 XNA motor de jogo

Até este ponto nenhum código foi inserido, mas para se poder começar a codificar é necessário conhecer as principais funções do XNA que estão na classe Game.cs. O motor à partida contém seis funções que representam os passos necessários do jogo. As funções são:

```
public Game1()

protected override void Initialize()

protected override void LoadContent()

protected override void UnloadContent()

protected override void Update(GameTime gameTime)

protected override void Draw(GameTime gameTime)
```

Estas funções já são criadas juntamente com o novo projecto de XNA. A primeira função é apenas o construtor do jogo que quando chamada representa o início do ciclo infinito. Esta inicializa automaticamente o Sistema gráfico para o jogo.

A função Initialize() destina-se a inicializar conteúdos não gráficos e alguns serviços que o utilizador possa necessitar, por exemplo será aqui que se irão inicializar (gerar) as bibliotecas de sons para o jogo.

A seguinte função, LoadContent() destina-se a introduzir o conteúdo gráfico necessário ao jogo; por exemplo para um jogo são necessárias texturas e é aqui que irão ser inicializadas.

A função UnloadContent(), é o inverso da anterior, caso um tipo de textura deixe de ser necessário poderá ser retirada de memoria usando esta função.

Estas duas funções são as principais ligações ao Content Pipeline, que tal como já foi dito é a componente de gestão de conteúdos do XNA Game Studio. O mais importante a referir acerca deste gestor é que sem o programador se preocupar em ler imagens de diferentes tipos ou gerir memoria para cada uma delas, o Content Pipeline já faz isso sozinho, facilitando assim o desenvolvimento do jogo.

É de notar que estas funções só são chamadas, normalmente, uma vez no jogo, ou seja quando o jogo é inicializado estas funções são chamadas automaticamente. A razão é simples, só é necessário dizer onde estão as texturas uma vez, ou seja pô-las em memória. Uma vez isto completado pode o jogo começar utilizando as funções seguintes.

A função Update (`GameTime gameTime`), destina-se literalmente à modificação do estado em cada frame, aqui será introduzido todo o gameplay do jogo, o estado dos objectos e as suas interacções. No caso do XNA este tenta manter sempre uma velocidade de frame de 60 frames por segundo, o que representa que a cada segundo que passa a função Update é chamada 60 vezes, tal como a próxima função de Draw.

A função Draw (`GameTime gameTime`), destina-se ao desenho dos estados, ou seja desenha 60 vezes por segundo todo o ecrã e os objectos do jogo.

Esta noção de Frames por Segundo (FPS), representa o movimento do jogo, na relalidade todo os objectos do jogo estão estáticos, mas basta que durante esses 60 frames por segundo os façamos aparecer em locais diferentes que, a noção que o homem tem é de movimento, exactamente como acontece com os filmes.

5.8.2. Inicializar componentes

Antes de se começar a programar o gameplay é necessário inserir os objectos do jogo, no caso deste teremos de inserir os fundos (background), as texturas das naves, das balas e mais algumas de apoio. O processo e fácil. Primeiro teremos de adicionar as imagens das texturas ao projecto. Para isto basta ir ao Soluction Explorer do jogo e adicionar um "Existing Item…" no Content do jogo tal como está exemplificado na imagem #.

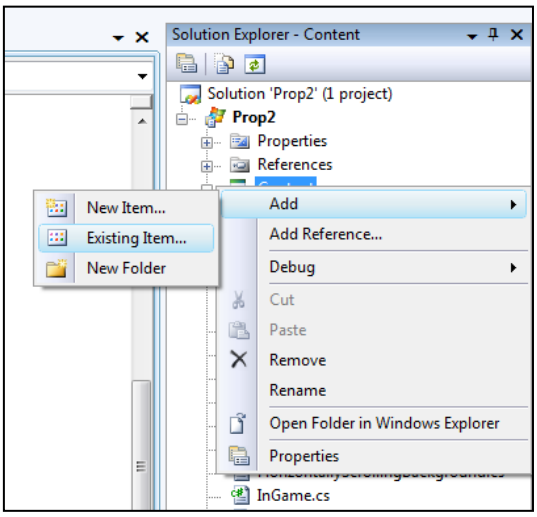


Figura 8 Adicionar Conteúdos

Este processo terá de ser repetido até todos os conteúdos estarem inseridos no jogo. Para que a apresentação do código seja melhor poderá dividir os conteúdos por pastas, criando novas pastas no "New Folder".

Uma vez que todas as texturas estão inseridas ter-se-á de as inicializar no código. Primeiro é necessário ter uma variável do tipo Texture2D e três Vector2 para guardar a posição da nave.

```
Texture2D NOME;
Vector2 Position;
Vector2 Direction;
Vector2 Speed;
```

Esta variável servirá para armazenar a textura, neste momento já se obteve um dos objectos do jogo, como o jogo será em 2D a textura pode ser utilizada como a nave

em si, só faltará ir à função LoadContent() e inseri-la na memoria do jogo. A posição, a velocidade e a direcção serão usadas mais tarde para poder movimentar a nave.

Tal como foi demonstrado a função LoadContent() introduz o conteúdo gráfico do jogo, este conteúdo será guardado no Content Manager do XNA

```
NOME = Content.Load<Texture2D>("numberone_right");

Position = new Vector2(START_POSITION_X, START_POSITION_Y);

Direction = Vector2.Zero;

Speed = Vector2.Zero;
```

O Content Manager é a ferramenta que foi descrita anteriormente por Content Pipeline, aqui será gerida a memoria necessária para tratar a imagem.

Neste momento a textura já está no jogo como um objecto, agora só falta o que este vai fazer e como o desenhar. Seguindo o GDD criado, sabe-se que a nave que este objecto representa irá movimentar-se (voar), no sentido dos pontos cardeais. E ter-se-ão de atribuir exactamente estes movimentos. Para isto ser possível é necessário compreender como é orientado o mundo virtual do XNA.

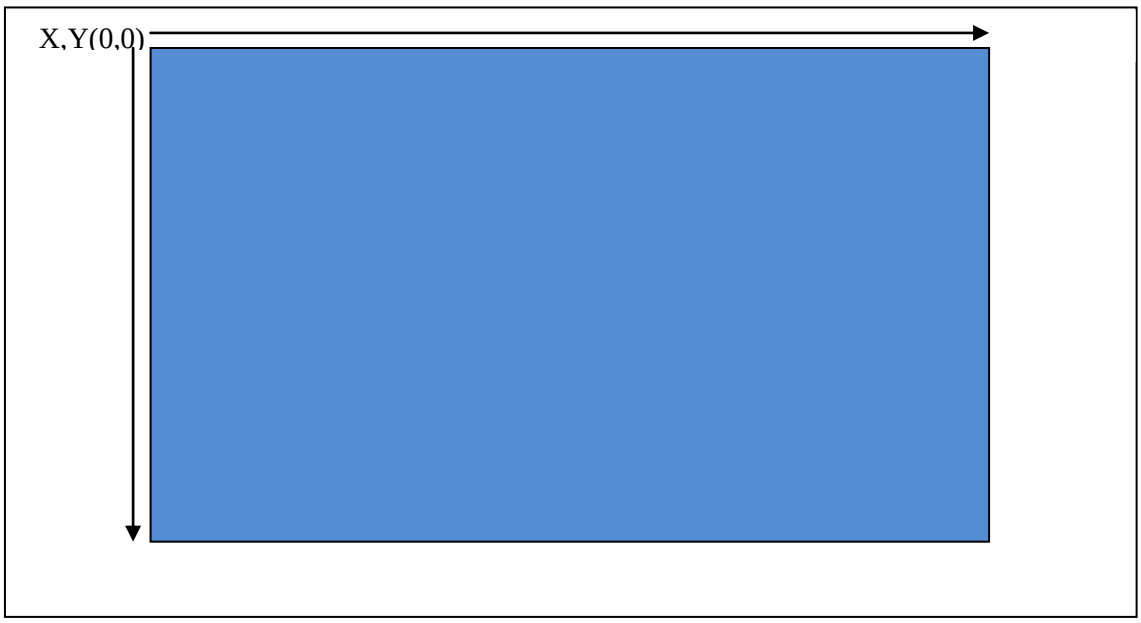


Gráfico 1 Sistema de Coordenadas

O XNA em 2D contém um sistema de coordenadas X e Y que começam em zero no canto superior esquerdo e aumentam a medida que se deslocam para o canto inferior direito, tal como está demonstrado no gráfico 1.

Para inserir a nave no jogo será necessário criar algumas constantes para ajudar,

```
const int START_POSITION_X = 125;

const int START_POSITION_Y = 150;

const int SHIP_SPEED = 160;

const int MOVE_UP = -1;

const int MOVE_DOWN = 1;

const int MOVE_LEFT = -1;

const int MOVE_RIGHT = 1;

const int MAX_DOWN = 550;

const int MAX_RIGHT = 750;
```

No princípio começa-se com a posição inicial da nave, tanto em X como em Y. A seguir atribui-se 160 a velocidade da nave, este número não é mensurável mas será usado mesmo para, usando cálculos matemáticos simples, calcular a posição da nave a cada frame. As quatro variáveis seguintes servem para modificar a direcção na qual a nave vai. Este processo está apresentado no gráfico 2. As duas últimas variáveis representam o valor máximo de X e Y para onde a nave se pode deslocar, obrigando a nave a não sair do campo de visão (tamanho do ecrã).

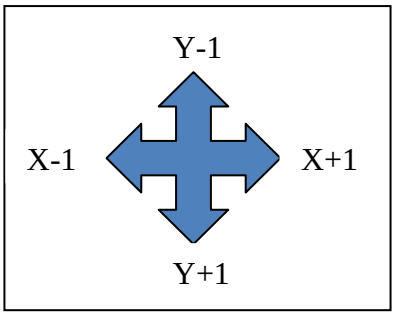


Gráfico 2 Modificação de coordenadas

5.8.3. Desenhar Componentes

Uma vez os componentes inicializados será necessário desenhá-los. Na função Draw() ir-se-á usar o SpriteBatch para o objectivo. O SpriteBatch é uma classe do XNA que quando é usado garante que todos os sprites (imagens) que forem utilizados terão as mesmas características em relação ao jogo. No inicio do código da classe Game.cs verifica-se que o SpriteBatch já esta inicializado.

```
SpriteBatch spriteBatch;
```

Agora só teremos de usar esta variável para desenhar no jogo. Na função draw teremos de realizar três passos. Iniciar o SpriteBatch, desenhar a nave e fechar o SpriteBatch. O código para realizar estas tarefas é o seguinte.

```
        spriteBatch.Begin();

        spriteBatch.Draw(NOME,new
Vector2(START_POSITION_X,START_POSITION_Y),Color.White);

        spriteBatch.End();
```

Deste código só será necessário explicar a linha do draw pois o Begin e o End são o iniciar e o fechar. Na segunda linha já se está mesmo a desenhar a nave. O NOME é a textura que esta a ser desenhada, o `new Vector2` vai ser a posição inicial da nave e a `Color.White` está no código porque representa a modulação das cores, que quando definida como branca apresentará todas as cores do espectro, ficando com as cores originais da imagem. Neste momento se o jogo fosse executado apareceria a imagem da nave dentro do ecrã do jogo, mas com o seu tamanho original.

Uma coisa a ter em conta quando se está a desenhar os objectos no ecrã é que as coordenadas dos objectos também seguem o mesmo sistema do ecrã do jogo, o que significa que o canto superior esquerdo da imagem é, para a própria, o ponto X,Y (0,0) como se demonstra no gráfico 3.

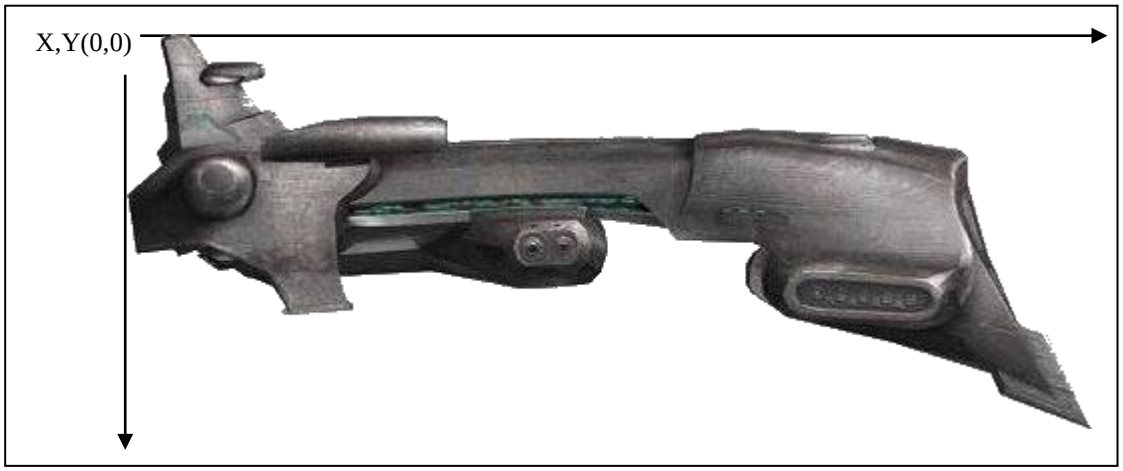


Gráfico 3 Coordenadas de um Sprite

Neste momento se o jogo fosse corrido, ir-se-ia ver uma imagem quase do tamanho do ecrã devido ao tamanho da imagem em si. De modo a não existir este problema, ter-se-á de modificar a escala da nave, mas para isso será necessário alterar o método de desenho da nave para a seguinte linha

```
spriteBatch.Draw(NOME,Position, new
     Rectangle(0,0,NOME.Width,NOME.Height),Color.White,0.0f,
     Vector2.Zero,0.3f,SpriteEffects.None,0);
```

Com esta nova linha de código, a imagem aparecerá na mesma no ecrã, mas agora com uma escala inferior em 0.3, o que significa cerca de um terço o seu tamanho original. A posição também foi alterada de modo a poder ser modificada posteriormente.

Para a função Draw() ter a possibilidade de usar escalas é necessário inserir mais uma série de coisas, uma vez que ficou quase tudo a zero de modo a não alterar nada que não se queira. Foi acrescentado o rectângulo que tem os limites da imagem desde o ponto (0,0) até ao ponto máximo da imagem (Max_Width, Max_Height). A seguir a cor, foi acrescentada a rotação, que no caso é zero, a seguir foi acrescentada a origem da imagem a zero mais uma vez, a escala e para acabar os efeitos da imagem e o nível a que será desenhada. Destes dois últimos, os efeitos estão desligados e o 0 representa o nível a que a nave vai ser desenhada.

5.8.4. Update do Gameplay

Uma vez a nave desenhada no ecrã, falta dar-lhe movimento, senão estaríamos a olhar apenas para uma imagem estática. Para esta finalidade ter-se-á de utilizar a função Update(). Aqui poderemos testar o teclado, rato ou gamepad e de acordo com o resultado alterar o estado da nave. Neste exemplo ir-se-á apenas usar o teclado mas facilmente se poderão encontrar dados de como usar os outros na comunidade XNA.

Primeiro é necessário saber o estado do teclado a cada momento, para isso ter-se-á de ter um variável que teste o teclado a cada ciclo do jogo e também uma segunda para se saber qual o estado do teclado no ciclo anterior.

```
KeyboardState PreviousKeyboardState;
```

Esta variável ira guardar o estado do teclado no ciclo anterior, uma vez na função Update(), criar-se-á a segunda que terá o estado actual.

```
KeyboardState aCurrentKeyboardState = Keyboard.GetState();
```

Obtendo o estado basta testá-lo, para saber quais as teclas que estão a ser pressionadas, para esta finalidade só será necessário usar o seguinte código.

```
Speed = Vector2.Zero;
Direction = Vector2.Zero;
```

```
KeyboardState aCurrentKeyboardState = Keyboard.GetState();

if (aCurrentKeyboardState.IsKeyDown(Keys.Left) == true)

{

    if (Position.X > 0)

    {

        Speed.X = SHIP_SPEED;

        Direction.X = MOVE_LEFT;

    }

    else if (Position.X < 0)

    {

        Speed.X = 0;

        Direction.X = 0;

    }

}

Position += Direction * Speed *
   (float)gameTime.ElapsedGameTime.TotalSeconds;

PreviousKeyboardState = aCurrentKeyboardState;
```

Este código é fácil de compreender. Primeiro é necessário passar a zero a velocidade e direcção, obrigando assim a nave a só mexer quando se pressionar uma tecla. O restante código é utilizado para testar a tecla seta da esquerda do cursor e caso esteja pressionada, atribuir velocidade e direcção adequada. No caso de a tecla ser pressionada ter-se-á de testar a posição em X (valor horizontal) para ter a certeza que a nave se encontra dentro do ecrã, nesse caso poderá movimentar-se, senão simplesmente não se move. Antes de passar para o resto do código será necessário repetir o processo para as restantes teclas. No final mostra-se como é que é efectuada a transformação destas variáveis para modificar a posição da nave. À posição soma-se a multiplicação da velocidade pela direcção, por exemplo 160 * (-1) dá uma velocidade negativa, logo a nave move-se para traz, e volta-se a multiplicar pelo total de segundos que passaram de jogo, fazendo com que a nave se mova sequencialmente e não saltando pixéis. Neste momento se o jogo fosse inicializado, iria aparecer a nave com um tamanho mais reduzido e com o uso dos cursores do teclado, mover-se-ia.

O jogo neste ponto ainda não está completo, mas esta secção apenas mostra a simplicidade que é criar objectos e usar os métodos dos XNA, o protótipo final, estará como anexo da dissertação, apenas é necessário mencionar que algum código do jogo foi retirado da comunidade XNA, que livremente disponibiliza para uso.

5.9. Projecto Scratch

O Scratch é uma nova linguagem de programação, desenvolvida no Massachusetts Intitute of Technology (MIT), que facilita a criação de histórias interactivas, animações, jogos, música e arte, da ainda a possibilidade de partilhar as criações na internet. Desenvolvido, originalmente, para ajudar crianças (8 anos e mais) a desenvolver capacidades de aprendizagem. Usando esta aplicação as crianças poderão experimentar e interiorizar capacidades importante do género de cálculos matemáticos, ideias informáticas, desenvolver a sua criatividade e lógica. De modo a criar um paralelismo com o XNA ir-se-á criar o mesmo pequeno exemplo que foi feito anteriormente. Para começar com o Scratch, basta ir ao site oficial e fazer download do programa que é livre. Uma vez instalado este apresenta automaticamente o ambiente de programação. Os Ambiente contem 4 secções principais, Secção de blocos, de scripting, de sprites e de jogo. As secções têm funções específicas a de blocos contem os movimentos, sons, aparência, caneta, controle, sensores, operadores e variáveis. Para replicar o jogo feito anteriormente bastará usar blocos de movimento e controle.

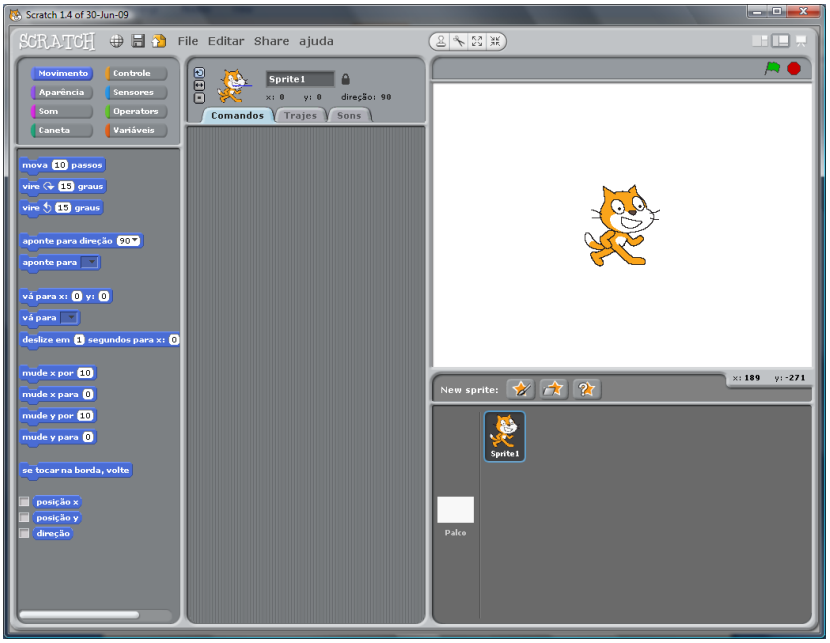


Figura 9 Ambiente inicial do Scratch

Inicialmente ter-se-á de apagar o sprite que vem com o Scratch, para isso basta clicar em cima do sprite do gato na secção de sprites com o botão da direita do rato e apagar. Após isto será introduzido o sprite da nave espacial utilizado anteriormente, para isso carrega-se no botão de "escolher novo objecto do arquivo".

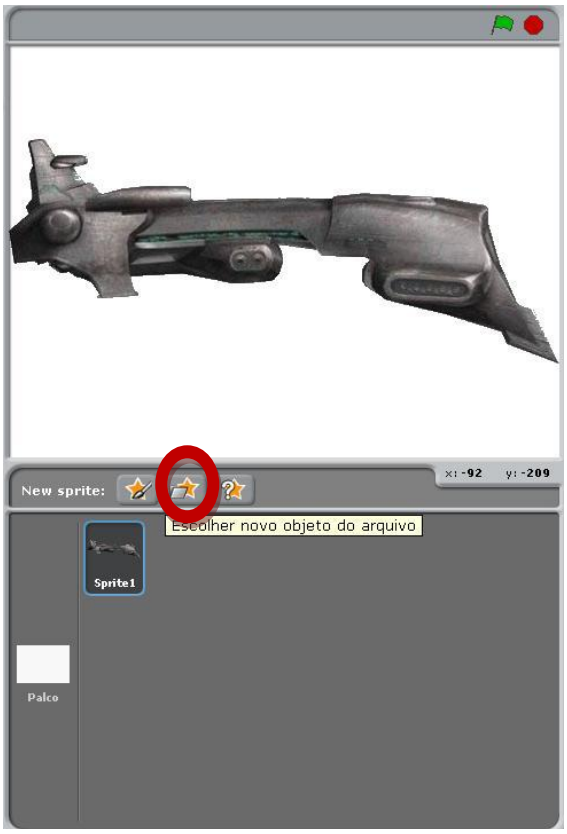


Figura 10 Inserção de sprites no Scratch

A imagem que se segue demonstra o botão que deve ser usado e o resultado da imagem depois de se inserir. Neste momento é necessário reduzir o sprite pois está do tamanho do espaço de jogo. Para isso basta usar o botão que se encontra por cima da imagem para reduzir a imagem ao tamanho pretendido.

Neste momento só falta criar os movimentos para a nave, de modo a andar para a direita, esquerda, cima e baixo. Para isso será necessário introduzir o bloco de movimento "mude x por [10]" este bloco fará que a nave anda para a direita, na direcção do X, 10 pixeis, para se usar as outras direcções basta subtrair 10 pixeis noutro bloco igual a este e para a direcção do Y basta usar o bloco "mude y por [10]".

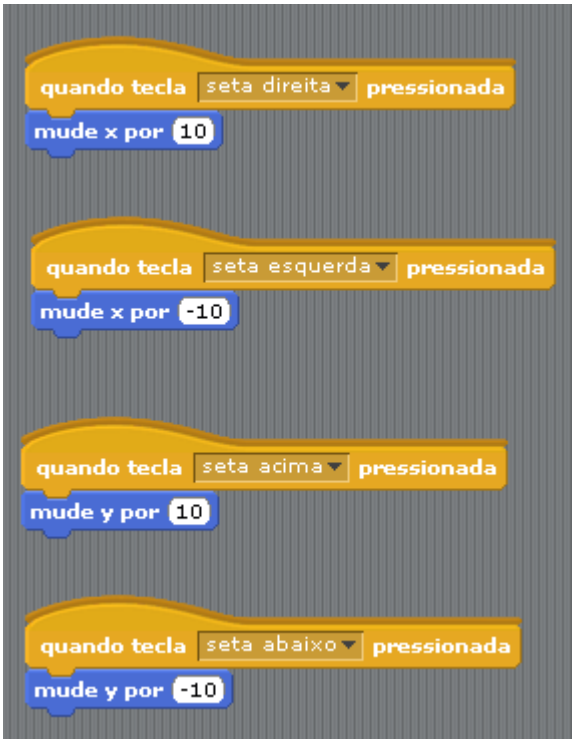


Figura 11 Blocos de script do jogo em Scratch

Com os movimentos introduzidos, falta dizer qual tecla é

que chama o bloco de movimento, para isso basta introduzir em cima de cada bloco de movimento um bloco de controlo “quando tecla [space] pressionada” e mudar o campo space pela tecla pretendida. No caso do jogo foram utilizadas as teclas dos cursores.

Com estes passos executados a nave já se movimenta igualando o que foi feito no XNA.

6. Estudo Empírico

6.1. Objectos de Estudo

Visto que a ideia de programação de jogos nem sempre é bem acolhida pela comunidade, foi idealizado um estudo para testar o interesse dos alunos. O estudo empírico realizado foi baseado num inquérito por questionário, aplicado na Web para os alunos de Engenharia Informática da Universidade Fernando Pessoa, no qual foi obtida uma amostra de 70 respostas num universo de pouco mais de 100 alunos. Foram escolhidos os alunos de Engenharia Informática devido ao intuito do questionário; de obter uma ideia real do interesse que os alunos do ramo de informática demonstram em relação à introdução da programação de jogos de computador no ensino superior.

O aluno de um curso superior no ramo da informática detém uma boa visão do plano de estudo por que está a passar e também uma ampla ideia dos gostos e objectivos dos seus colegas, assim é o melhor objecto de estudo para obter uma previsão sobre o interesse desses alunos na programação de jogos para o ensino superior.

6.2. Motivação

Um curso que seja envolvido no mundo da computação de jogos pode, não só ser uma mais-valia para os cursos de informática, a começar pela obtenção de uma maior atenção e interesse dos seus alunos e terminando no desenvolvimento de grandes profissionais que criarão e aplicarão tecnologias, hoje em dia impensáveis. Durante um curso de Engenharia Informática o aluno é bombardeado com uma série de ferramentas e processos / metodologias para o desenvolvimento de aplicações. Mas estas aplicações nem sempre são motivantes para o aluno. A inserção de projectos que possam motivar mais o aluno a realizá-los, não só com a ideia de aprender mas também pelo gosto de o concretizar, desenvolverão motivações e capacidades fazendo com que este aluno possa demonstrar uma destreza muito superior e assim superar-se consecutivamente para atingir melhores e maiores resultados.

A inserção da programação de jogos como “projectos efectivos de disciplinas” e mesmo de “cursos superiores” já é uma realidade em muitas universidades pelo mundo fora, tendo já sido realizados estudos que demonstraram o interesse dos alunos por esses cursos, [Zyda 2008] [Ferguson 2007].

Em Portugal só recentemente é que apareceram os primeiros cursos de programação de jogos ao nível superior, um exemplo é o curso de Design de Jogos Digitais do Instituto Politécnico de Bragança [pagina oficial do Instituto Politécnico de Bragança]. Mas já anteriormente existiram alguns institutos e núcleos que demonstraram algum interesse, tal como o Instituto Politécnico de Portalegre que inseriu algumas disciplinas de concepção de jogos no plano curricular do curso de Engenharia Informática e o Núcleo Estudantil de Computação Gráfica da Faculdade de Engenharia do Porto. [pagina oficial do Instituto Politécnico de Portalegre] [pagina oficial do Núcleo Estudantil de Computação Gráfica]

O desafio é o de criar um plano de estudos dedicado ao desenvolvimento de jogos, que não depurasse o aluno das características essenciais de que este necessita para ser um Engenheiro Informático mas que em simultâneo o motive a realizar o curso superando-se em cada momento.

Com o passar do tempo o desenvolvimento de jogos está a ser integrado na comunidade académica, pelo mundo fora, como um meio de aprendizagem de confiança. Como tal as grandes Universidades criaram cursos com esse objectivo. Cursos orientados ao desenvolvimento da programação de jogos estão implementados em muitas universidades por todo o mundo e em grande expansão.

Um curso de desenvolvimento de jogos pode ser extremamente caro envolvendo várias ferramentas de custos altos, mas a Microsoft, lançou para o mercado uma ferramenta que para além de ser livre, pode facilmente ser usada para o ensino. Esta ferramenta é o «XNA Game Studio» que integra grande parte das funcionalidades que são hoje ensinadas nos cursos de Engenharia Informática simplificando assim a sua inserção.

O XNA é a Framework.NET para o desenvolvimento de jogos, tanto para a plataforma de Windows como para a XBOX 360 ou ZUNE. Tanto o «XNA Game Studio» como o C# Express estão livres para download na Microsoft. Estas ferramentas permitem assim o estudo de desenvolvimento de jogos para departamentos com baixos recursos, facilitando a sua inserção no currículo de um Curso de Engenharia Informática.

6.3. Método de Recolha

O método de recolha usado para o questionário, foi o da introdução do mesmo numa página Web usando o LimeSurvey [pagina oficial do LimeSurvey], uma aplicação de criação e gestão de questionários livre. A aplicação foi inserida num espaço Web disponibilizado pelo ES-CEFOC – Estudo de Sondagens e Centro de Formação Contínua da Universidade Fernando Pessoa. Com o apoio da coordenação do Curso de Engenharia de Software o questionário foi divulgado, possibilitando atingir o maior número possível de alunos.

6.4. Questionário - Análise de Resultados

Iniciar-se-á por resumir as perguntas e conclusões retiradas do questionário após uma primeira revisão. As primeiras perguntas tinham como objectivo saber quais as disciplinas que os alunos consideravam serem necessárias para a realização de um bom curso de Engenharia Informática. As disciplinas mais indicadas foram as de Algoritmos, de Programação e de Engenharia de Software. Após estas perguntas foi averiguado o interesse dos alunos pelos jogos e pela própria programação dos mesmos. Sem grande novidade resultou numa opinião quase unânime em relação ao gosto por jogos bem como igualmente ao interesse pela sua programação.

Prosseguindo foram inquiridas as motivações que levam um aluno a seguir para o ensino superior e se achava que se o curso de Engenharia Informática contivesse programação de jogos no seu plano poderia ser um factor de motivação ainda maior para a escolha do curso pelo aluno.

A importância da existência destes cursos também foi questionada aos alunos, que por sua vez responderam que neste momento seria urgente existirem cursos do género. Neste caso a resposta com mais escolha foi a que correspondia à promoção de uma melhor formação na área.

Por curiosidade foi inquirido aos alunos quais as linguagens de programação de jogos conheciam, e o resultado foi principalmente as linguagens de programação que são ensinadas actualmente no curso de engenharia, o Java e o C, apesar de estas não serem linguagens dedicadas à programação de jogos. Em segundo lugar aparecem duas

que sim já são dedicadas apenas à programação de jogos, o OpenGL e o Microsoft XNA.

Para finalizar foi perguntado quais seriam as maiores dificuldades para a definição de cursos superiores dedicados à programação de jogos. Neste caso as respostas foram pouco expressivas, pois situaram-se bastante nos valores médios entre muito difícil e nada difícil.

Aprofundando mais as respostas do questionário verificar-se-á o verdadeiro interesse destes alunos pela programação de jogos de computador, como se verá a seguir.

Na primeira questão foi inserido um quadro no pressuposto de obter a opinião do aluno quanto ao tipo de conhecimento que achava crítico para um engenheiro informático. Este quadro oferecia um conjunto de respostas que iam desde o nível 1 (Pouco Critico) até ao nível 7 (Muito Critico) inserindo também a hipótese de não responder. Para que os alunos focassem melhor as suas respostas foram usados nomes de matérias geralmente dadas no curso. Sendo no quadro a primeira pergunta enumeraram-se as alternativas em alíneas da seguinte forma:

1.1 Tecnologias de Sistemas Operativos
1.2 Programação de computadores (Algoritmia e programação)
1.3 Programação orientada aos objectos
1.4 Redes de computadores e comunicação de dados
1.5 Sistemas de informação
1.6 Multimédia
1.7 Base de Dados
1.8 Engenharia de Software
1.9 Outros. (Por favor, especificar)

Foi sem grande surpresa que se verificou que as matérias que os alunos acharam mais críticas para o curso fossem Programação de computadores (Algoritmos e Programação) e Engenharia de Software tal como o demonstram os seguintes gráficos:

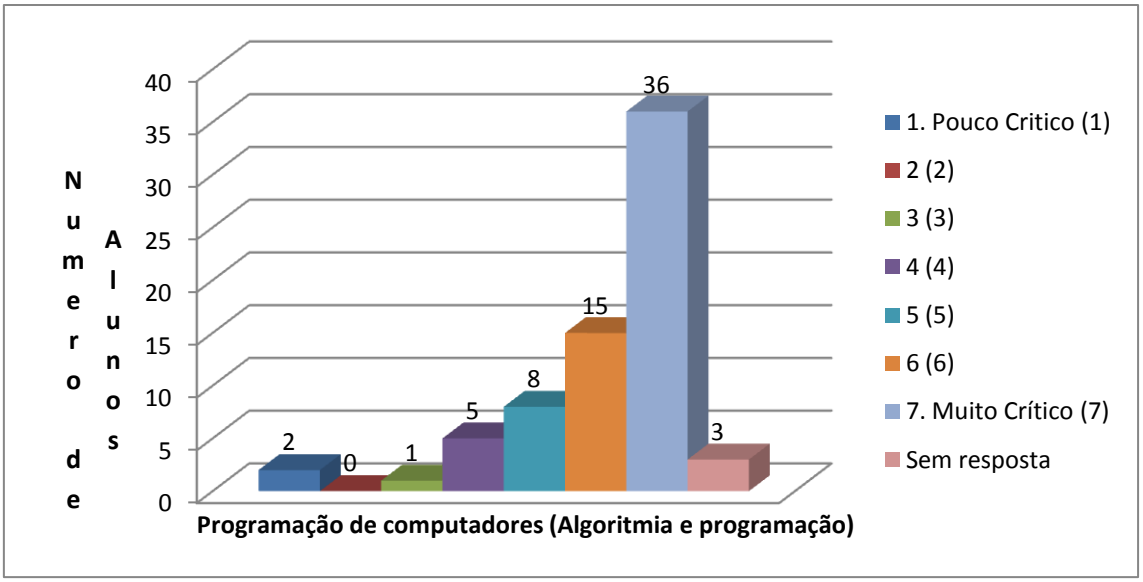


Gráfico 4 Que tipo de conhecimento considera crítico para um Engenheiro Informático? – Programação de computadores

Este gráfico representa o número de alunos que achou muito crítico a aprendizagem de Programação de computadores. Em termos de percentagem verifica-se que 51.43% dos alunos acredita que esta matéria é a mais importante para o curso.

O gráfico seguinte representa a segunda área que os alunos acreditam ser mais crítica para um engenheiro informático, a disciplina de Engenharia de Software.

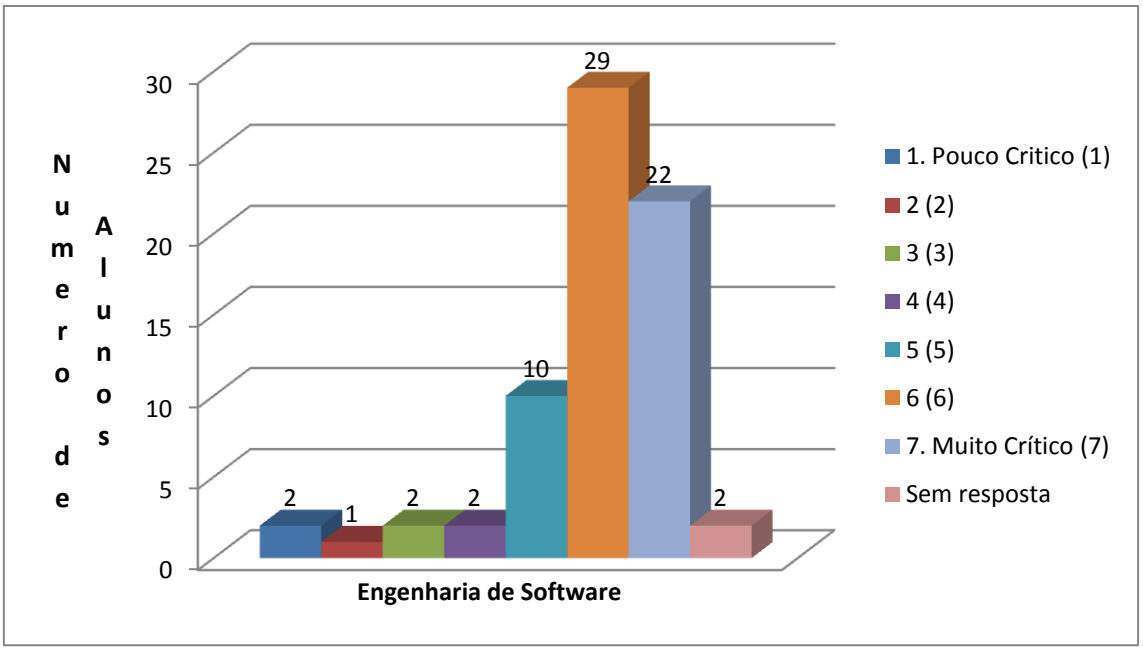


Gráfico 5 Que tipo de conhecimento considera crítico para um Engenheiro Informático? – Engenharia de Software

Apesar de uma grande quantidade dos alunos não considerar esta matéria como muito crítica, não deixa de ser uma escolha bastante importante para o curso.

Nas seguintes perguntas, da segunda à quarta, indagava-se sobre os gostos dos alunos e a curiosidade que estes detinham pela indústria dos jogos. As seguintes perguntas foram expostas juntamente com as opções possíveis de Sim, Não e Sem Resposta.

2. Gosta de jogos de computador?
3. Gostava de saber como funcionam (e são programados) os jogos?
4. Gostaria de saber criar os seus próprios jogos?

Obtiveram-se repostas para as 3 perguntas que rondavam os 85% de Sim. Este valor não é surpreendente visto se estar a falar de alunos de informática. Os gráficos seguintes demonstram exactamente os resultados das perguntas.

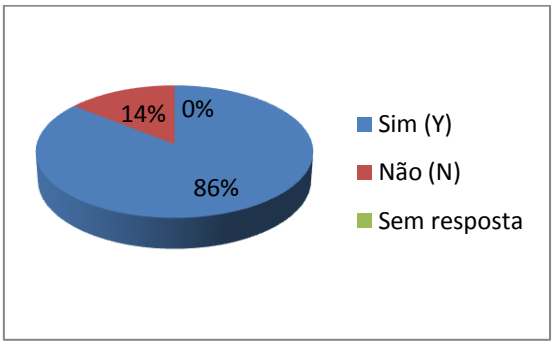


Gráfico 6 Gosta de jogos de computador?

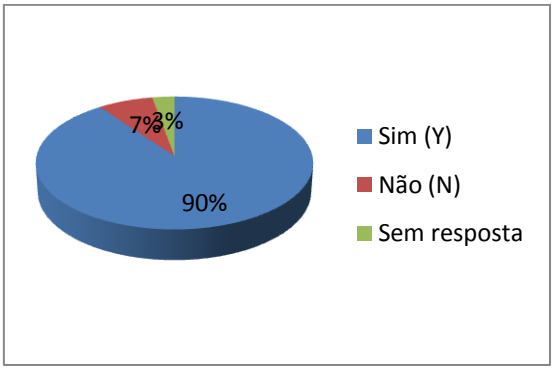


Gráfico 7 Gostava de saber como funcionam (e são programados) os jogos?

Estes resultados demonstram sem sombra de dúvida o interesse que um aluno do ensino superior do ramo da Informática tem para com esta matéria. Poder-se-ia criar uma solução para apoiar os alunos a desenvolver jogos de computador.

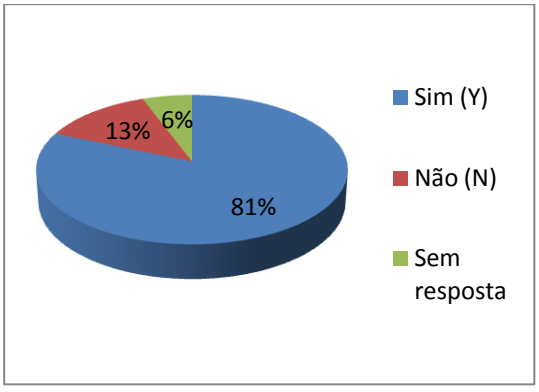


Gráfico 8 Gostaria de saber criar os seus próprios jogos?

A motivação dos alunos para o ingresso no ensino superior também foi inquirida neste questionário e o quanto motivado estaria o aluno caso o curso fosse de programação de jogos de computador, deste modo foram feitas as seguintes questões:

5. Considera que os jogos de computador são um factor de motivação para o ingresso de alunos no ensino superior no ramo da informática?
6. Acredita que a motivação é um factor essencial para o progresso do aluno no ensino superior?
7. O que poderá motivar mais o progresso de um aluno?

Mais uma vez os resultados não foram surpreendentes pois os alunos voltaram a responder Sim com bastante frequência, isto para as perguntas 5 e 6, no caso da pergunta 7 foi-lhes exposto as seguintes quatro opções de escolha.

- Jogar
- Desenvolver / programar
- Concepção / história do jogo
- Sem Resposta

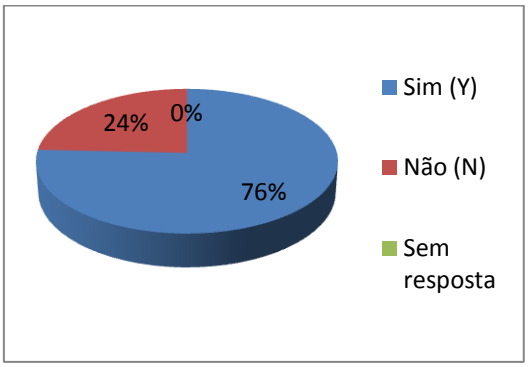


Gráfico 9 Considera que os jogos de computador são um factor de motivação para o ingresso de alunos no ensino superior no ramo da informática?

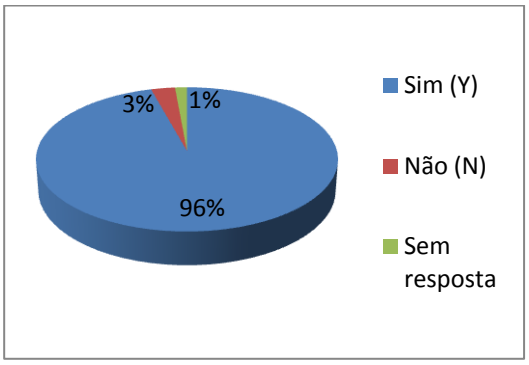


Gráfico 10 Acredita que a motivação é um factor essencial para o progresso do aluno no ensino superior?

Para as perguntas 5 e 6 as respostas são bastante esclarecedoras, a maioria dos alunos considera que a motivação é um factor essencial para o progresso do aluno no ensino superior e também, embora em menor número, consideram que os jogos podem ser um factor de motivação para o ingresso no ensino superior.

No próximo gráfico confirma-se o interesse dos alunos pela programação e que não estão a tirar o curso para jogar ou contar histórias, mas sim para programar e desenvolver software.

Neste caso 70% dos inquiridos responderam Desenvolver / programar.

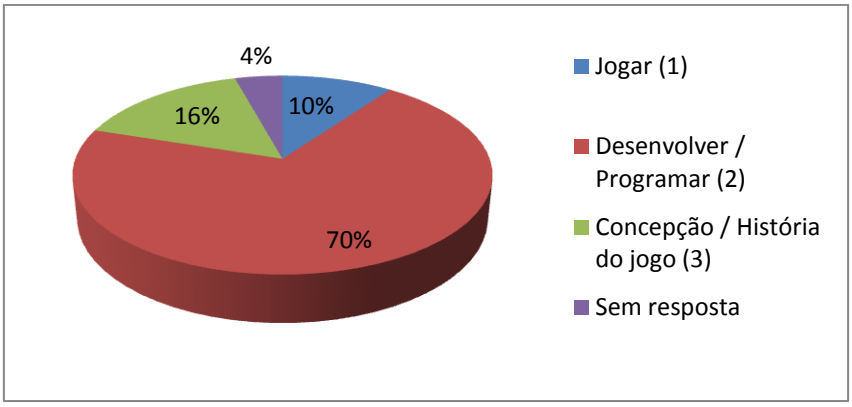


Gráfico 11 O que poderá motivar mais o progresso de um aluno?

As cinco perguntas seguintes pretendem demonstrar o que os alunos pensam de um curso baseado em jogos de computador e no caso deste existir onde seria melhor aplicado, ou seja em que nível de escolaridade seria inserido mais correctamente e se concordam que um curso destes poderia ter boas saídas profissionais.

8. Concorda com uma lógica de aprendizagem baseada na programação de jogos de computador no ensino superior?
9. Uma vez feita a opção profissional, de programador de jogos, considera que esta é uma profissão com futuro?
10. Sabendo que a programação de jogos utiliza as linguagens de programação, considera adequada a utilização de uma aprendizagem baseada na programação de jogos para uma disciplina de programação?
11. Sabendo que a programação de jogos necessita de metodologias avançadas de investigação e desenvolvimento poderá ser usada para disciplinas de engenharia de software ou análise de sistemas?
12. Em que nível de educação considera que a programação de jogos, pode ser inserida.

Mais uma vez as perguntas tem respostas de Sim e Não, à excepção da última que oferece quatro escolhas,

- Secundário (10º – 12º anos de escolaridade)
- Licenciatura.
- Mestrado.
- Não sabe.

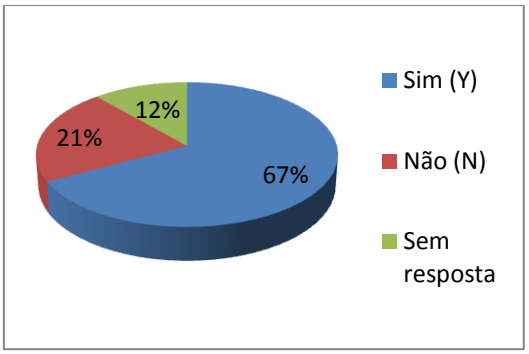


Gráfico 12 Concorda com uma lógica de aprendizagem baseada na programação de jogos de computador no ensino superior?

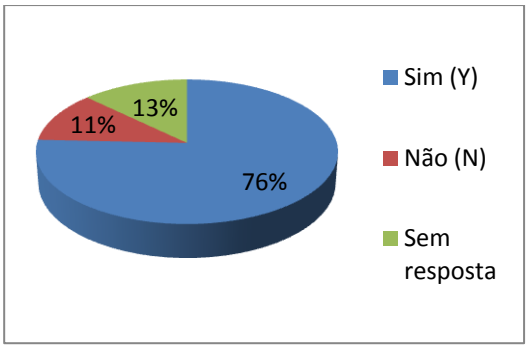


Gráfico 13 Uma vez feita a opção profissional, de programador de jogos, considera que esta é uma profissão com futuro?

Nestas respostas, regista-se que os alunos acreditam no uso de uma aprendizagem baseada em jogos e que crêem poder ter boas saídas profissionais quando acabarem o curso.

Curiosamente as perguntas 10 e 11 foram feitas de modo a obter uma ideia se os alunos concordariam com a inserção de matérias relacionadas com a programação de jogos nas disciplinas de programação e de engenharia de software, o curioso foi o facto de os alunos acharem que estas são as disciplinas mais críticas para o curso e concordarem com a inserção destas matérias nas disciplinas.

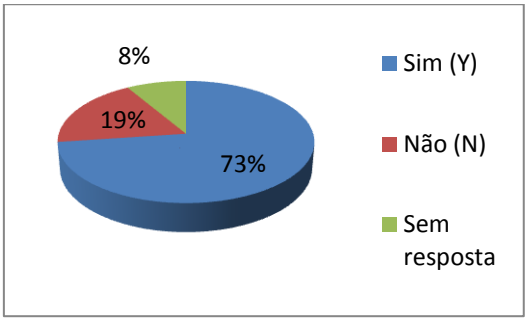


Gráfico 14 Sabendo que a programação de jogos utiliza as linguagens de programação, considera adequada a utilização de uma aprendizagem baseada na programação de jogos para uma disciplina de programação?

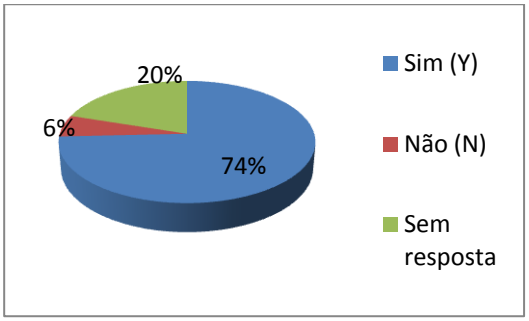


Gráfico 15 Sabendo que a programação de jogos necessita de metodologias avançadas de investigação e desenvolvimento poderá ser usada para disciplinas de engenharia de software ou análise de sistemas?

Como os gráficos demonstram os alunos concordam com a utilização de uma aprendizagem baseada na programação de jogos de computador para as disciplinas de Programação e Engenharia de Software ou Análise de Sistemas.

De todas as respostas até agora revistas, pode-se desde já concluir que um curso de Engenharia Informática orientado para a programação de jogos de computador seria bem visto pelos alunos.

Falta agora saber em que altura da escolaridade, consideram os alunos que o curso deveria ser inserido. Esta pergunta é a 12 do questionário e cria um dilema. Os alunos dividiram-se entre a licenciatura e o mestrado, criando assim um problema. O resultado é demonstrado no seguinte gráfico:

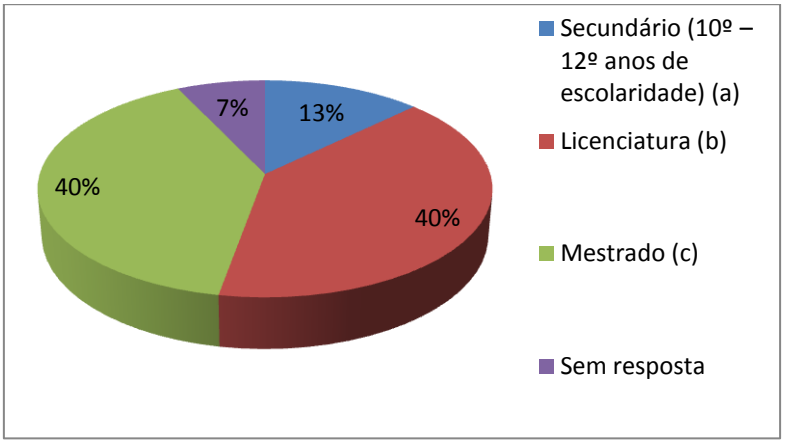


Gráfico 16 Em que nível de educação considera que a programação de jogos, pode ser inserida

Como o gráfico demonstra a licenciatura e o mestrado receberam ambos 40% das respostas dos alunos, passando assim a opção da inserção do curso para a instituição que o oferece. Apenas se pode concluir que o curso pertence ao ensino superior.

Na pergunta 13 foi inserido um novo quadro de respostas por níveis. Nele pretendia-se saber o que os alunos achavam sobre a importância da existência de cursos dedicados à programação de jogos. Para tal foram inseridas as seguintes propostas:

1. Novos ramos de desenvolvimento e novas licenciaturas
2. Criação de cursos superiores mais dedicados à programação de jogos
3. Promover a formação na área
4. Aplicar esta tecnologia noutros ambientes e processos de negócio
5. Desenvolver novas características e competências para os estudantes

6. Inovar produtos, procedimentos e serviços

Mais uma vez foram usados os níveis de 1 (Pouco critico) a 7 (Muito Critico) e Sem resposta para finalizar.

Os resultados obtidos mostram que as opções 5, 6 e 7 somadas são significativamente mais volumosas que as quatro primeiras, demonstrando assim o interesse e a importância que os alunos atribuem à existência dos cursos em análise, como o apresenta o seguinte gráfico:

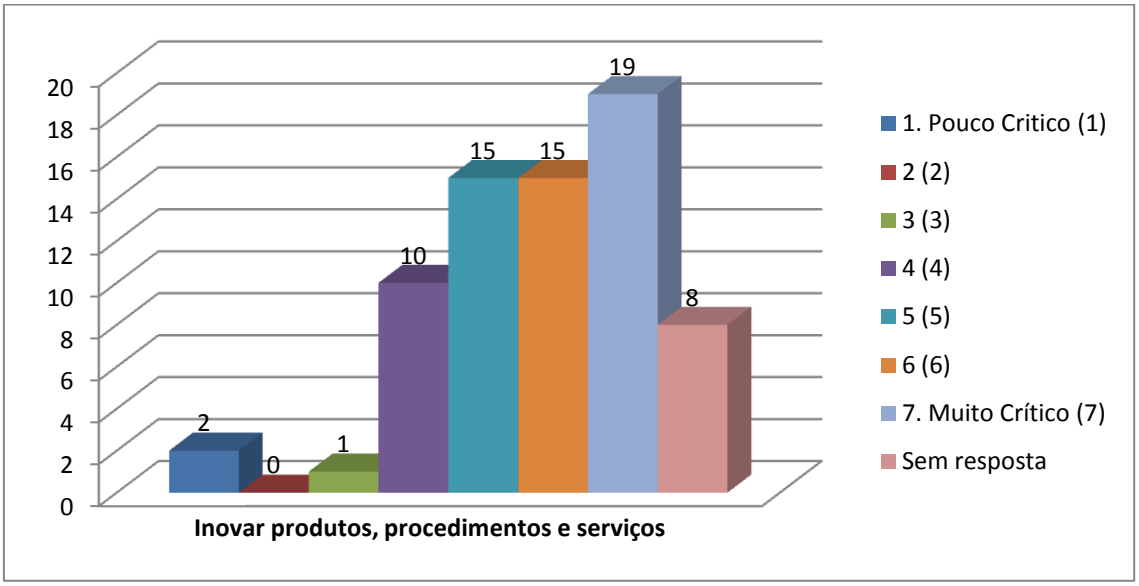


Gráfico 17 Que nível de importância atribui à necessidade de existirem cursos dedicados à programação de jogos - Inovar produtos, procedimentos e serviços

De modo a obter uma noção real das linguagens de programação que os alunos conheciam, foi inserida uma pergunta que apresentava algumas das linguagens de programação mais utilizadas para a programação de jogo, desta maneira poder-se-ia saber o que os alunos mais conheciam. O seguinte gráfico representa essa pergunta.

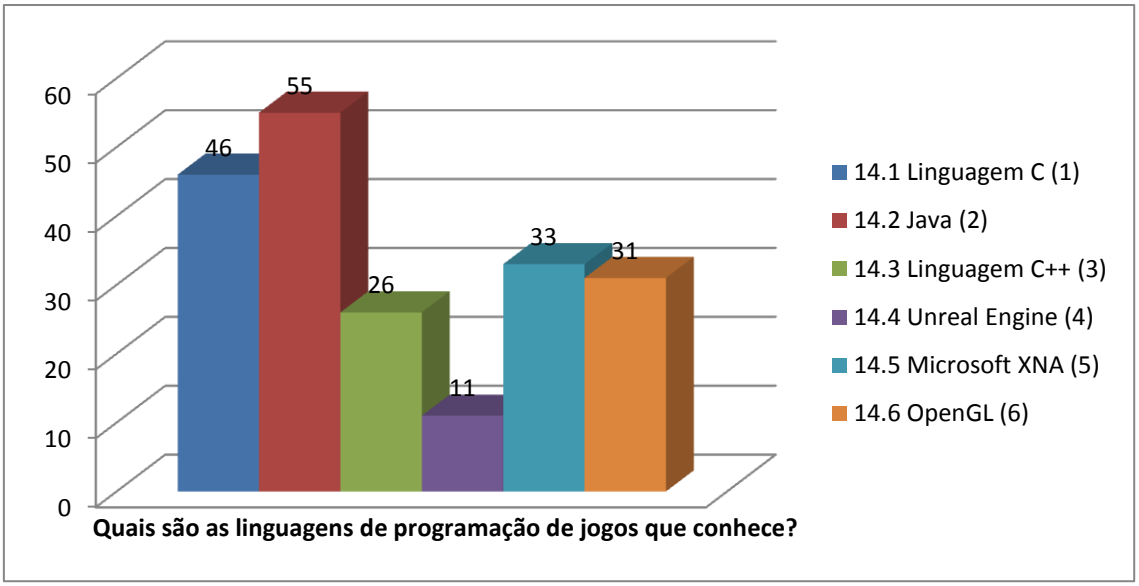


Gráfico 18 Quais são as linguagens de programação de jogos que conhece?

É relativamente sem grande surpresa que as linguagens mais conhecidas sejam o Java e o C, isto também porque para além de serem as matérias mais leccionadas nos cursos de Engenharia de Software, são as mais usadas no mundo inteiro.

A pergunta 15 do questionário, tenta perceber se os alunos compreendem os problemas que podem aparecer com a criação de cursos dedicados ao ensino de programação de jogos. Foi assim criado um quadro de respostas por níveis, mas que desta vez ia desde o nível 1 (Pouco difícil) até ao nível 7 (Muito difícil) e com a hipótese de não dar resposta como anteriormente. As perguntas eram as seguintes.

1. Aspectos culturais
2. Problemas de comunicação e exposição dos cursos
3. Aceitação do publico em geral
4. Carência de recursos humanos e competências na área
5. Infra-estruturas tecnológicas inadequadas
6. Carência de recursos financeiros
7. Tempo disponibilizado pelas partes envolvidas
8. Outros. (Por favor, especificar)

Neste caso os alunos orientaram as suas respostas principalmente para o nível 4 obtendo um conjunto de respostas muito medianas. Muito provavelmente esta pergunta não era adequada aos alunos, mas sim a coordenadores de curso e outros responsáveis.

7. Conclusão

O objectivo desta dissertação foi tentar identificar porque razão o desenvolvimento de jogos de computador não é realizado em grande escala em Portugal. Isto poderia ser justificado pela pouca oferta de cursos superiores dedicados a esta área. Esta situação sofreu uma alteração satisfatória, com a criação de dois cursos de ensino superior, direccionados para a temática de programação de jogos. O problema que se coloca agora é a não existência de um modelo de engenharia de software dedicado à programação de jogos. Com o desenvolvimento desta dissertação, procura-se solucionar esta falha expondo um modelo de software direccionado exclusivamente para a criação de jogos de computador. De modo a comprovar a metodologia, esta foi utilizada para criar um protótipo de um jogo de computador.

A principal preocupação no processo de implementação de um novo curso, ou área de interesse, é tentar discernir se esta terá ou não, interesse e receptividade por parte do público-alvo, neste caso, alunos a terminar a sua área de formação obrigatória. Para o presente trabalho não foi possível obter opiniões junto destes alunos, pelo que se resolveu realizar um questionário online aos alunos de Engenharia Informática, no qual teriam oportunidade de responder a determinas questões que avaliavam o seu interesse, sobre a possível inserção de programação de jogos no programa curricular.

Um aluno que se sinta estimulado com o desenvolver do curso, é com certeza um aluno melhor, mais dedicado. A motivação foi um factor importante quando foi executado o estudo empírico, pois era necessário saber qual a opinião dos alunos sobre a inserção de programas curriculares orientados aos jogos no curso de Engenharia Informática. Com a opinião dos alunos valoriza-se a necessidade destes programas, como foi provado, uma grande maioria de alunos concorda plenamente com esta abordagem reflectida nestes cursos.

A conclusão a que se chega após a avaliação de um conjunto significativo de opiniões é que, uma larga maioria, concorda e sentir-se-ia mais estimulada se o curso de Engenharia Informática incluísse uma valência de programação de jogos no seu currículo.

A conclusão final que se pode retirar desta dissertação, impulsiona à criação de mais cursos dedicados à programação de jogos, visto ser uma área com evidente futuro

e para a qual os alunos demonstram imenso interesse. Com a ajuda de um modelo de Engenharia de Software para o desenvolvimento de jogos de computador estes novos cursos seriam mais uma ferramenta para estimular os alunos, não sendo obrigados e seguir modelos convencionais, mas não retirando aos alunos os requisitos necessários para a criação de software.

Foram duas as limitações que se apresentaram durante o desenvolvimento desta dissertação: A primeira foi o grande conjunto de informação que existe nesta área, tanto ao nível do desenvolvimento de jogos como ao nível da Engenharia de Software. Não existe ainda um modelo de engenharia de software dedicado à área em questão, e existem inúmeros géneros de jogos. Esta limitação obriga então a criar um modelo generalizado, pois contem a necessidade de funcionar para qualquer tipo de jogo.

A segunda limitação foi a impossibilidade de contactar com alunos que estivessem a terminar o ensino obrigatório. Deste modo poder-se-ia realmente saber o seu verdadeiro interesse nesta área. Assim foram contactados alunos do primeiro ano de Engenharia Informática.

Uma perspectiva futura que se podia executar seria a utilização desta dissertação para o apoio à criação de um novo curso ou mestrado orientado ao desenvolvimento de jogos de computador.

8. Bibliografia

Baldwin, M. (2005). “Game Design Document Outline” Baldwin Consultant

Barker, F. S. (2006).”Visual C# 2005 Express Edition Starter Kit”, Editora Wrox

Beard, S.D., Ringo, L.A., Mader, B., Buchmann, E.H., Tanita, T. (2008). “Space Shuttle Landing and Rollout Training at the Vertical Motion Simulator”, American Institute of Aeronautics and Astronautics

Boehm, B. (1988).*"A Spiral Model of Software Development and Enhancement"* - IEEE Computer, vol.21

Chentanez, N., Alterovitz,R., Ritchie,D., Cho, L., Hauser,K.K., Goldberg, K., Shewchuck,J.R., O’Brien,J.(2009),“Interactive Simulation of Surgical Needle Insertion and Steering”. ACM Transactions on Graphics, Vol. 28, No. 3, Article 88

Creators Club de XNA http://creators.xna.com/ visitado em 30/10/2009

Dennis, M. R. (1996). “The Evolution of the Unix Time-sharing System”, Lucent Technologies Inc.

Doughty, M. (). “*COMPUTER GAME DEVELOPMENT EDUCATION AT UNIVERSITY*”, University of Lincoln, Brayford Pool, Lincoln, UK

edHeads Operação a um joelho – http://www.edheads.org/activities/knee/ - visitado 06/10/2009

Ferguson,E. Rockhold, B. Heck, B.(2007).”Video Game Development Using XNA Game Studio And C#.NET” Artigo ACM New York 115-118

GameMaker - http://www.yoyogames.com/make - visitado 24/11/2009

Grootjans, R. (2009), *“XNA 3.0 Game Programming Recipes: A Problem-Solution Approach”*, Editora Apress

Instituto Politécnico de Portalegre –http://www.estgp.pt/ensino/07_08/eijs/eijm.asp visitado em 23/03/2009

Kabashi, A. El-Saabi, H. (2006). *"Game Software Process"*, Tese.

Miles, R, (2008)." *Microsoft® XNA™ Game Studio 2.0: Learn Programming Now!"*, Editora Microsoft.

Nacke, L. (2005). *"Facilitating the education of game development",* Artigo

Núcleo Estudantil de Computação Gráfica – http://necg.fe.up.pt/necgsite/noticias visitado em 23/03/2009

Pagina oficial do curso de Design de Jogos Digitais do Instituto Politécnico de Bragança - http://djd.esact.ipb.pt/acesso.htm - visitado 26/10/2009

Pagina oficial do jogo Flight Simulator X - http://www.microsoft.com/games/flightsimulatorX/ visitado em 29/09/2009

Pagina oficial da plataforma para questionários online LimeSurvey - http://www.limesurvey.org/ visitado 28/12/2009

Pressman, R. S. (2006). *"Engenharia de Software"*, Editora Mac Graw Hill

Scratch - http://scratch.mit.edu/ - Visitado 17/12/2009

Second Life – http://secondlife.com/ - visitado 06/10/2009

Silva, J. Sedlak, J. (2008). "Building XNA 2.0 Games" Editora Apress

Software De Cirurgia Plástica Virtual 1.0 – http://www.photo-warping.com/pt-br - Visitado 06/10/2009

Sommerville, I. (1997), *"Software Engineering",* Editora Addison-Wesley

(2004). *"SWEBOK Guide to the Software Engineering Body of Knowledge"*, © IEEE – 2004 Version

Unreal Engine - http://www.unrealtechnology.com/ - visitado 24/11/2009

Wallace, S. Russel, I. Markov, Z. (2008)."Integrating Games and Machine in th Undergraduated Computer Science Classroom" Artigo ACM GDCSE'08

XNA Game Studio - http://msdn.microsoft.com/en-gb/xna/default.aspx - visitado 24/11/2009

Zyda, M. Lacour, V. Swain, C. (2008).”Operating a Computer Science Game Degree Program” Artigo ACM GDCSE’08

9. Anexos

9.1. Questionário – Estudo do perfil de um aluno para um curso superior de informática no âmbito da programação de jogos

Exmo(a). Sr(a).,

Este questionário foi desenvolvido com a finalidade de identificar e analisar os perfis dos alunos que ingressam num curso superior de informática, e que terão interesse num ramo de especialização em programação de jogos de computador. Como tal, ficaria muito grato se fizesse o favor de o preencher, respondendo a todas as questões com a máxima sinceridade. Trata-se de um questionário em que se garante o completo anonimato e a confidencialidade das respostas dadas pelos participantes.

O objectivo deste questionário é o de identificar os motivos que possam levar um candidato ao ensino superior a ingressar num curso de informática que integre um ramo de programação de jogos de computador. Neste contexto, as questões abrangem, em particular, os aspectos mais significativos de um curso que poderão motivar um aluno para o desenvolvimento de simulações virtuais interactivas, isto, de jogos de computador.

1. Que tipo de conhecimento considera critico para um Engenheiro Informático?

	Pouco crítico						**Muito crítico**
1.1 Tecnologias de Sistemas Operativos							
1.2 Programação de computadores (Algoritmia e programação)							
1.3 Programação orientada aos objectos							
1.4 Redes de computadores e comunicação de dados							
1.5 Sistemas de informação							
1.6 Multimédia							
1.7 Base de Dados							
1.8 Engenharia de Software							
1.9 Outros. (Por favor, especificar) - -							

2. Gosta de jogos de computador?
   a. □ Sim.
   b. □ Não.
   c. □ Não aplicável.
3. Gostava de saber como funcionam (e são programados) os jogos?
   a. □ Sim.
   b. □ Não.
   c. □ Não aplicável.

(continue na página seguinte, por favor)

4. Gostaria de saber criar os seus próprios jogos?
    a. □ Sim.
    b. □ Não.
    c. □ Não aplicável.
5. Considera que os jogos de computador são um factor de motivação para o ingresso de alunos no ensino superior no ramo da informática?
    a. □ Sim.
    b. □ Não.
    c. □ Não aplicável.
6. Acredita que a motivação é um factor essencial para o progresso do aluno no ensino superior?
    a. □ Sim.
    b. □ Não.
    c. □ Não aplicável.
7. O que poderá motivar mais o progresso de um aluno?
    a. □ Jogar
    b. □ Desenvolver/programar
    c. □ Concepção / história do jogo
8. Concorda com uma lógica de aprendizagem baseada na programação de jogos de computador no ensino superior?
    a. □ Sim.
    b. □ Não.
    c. □ Não aplicável.
9. Uma vez feita a opção profissional, de programador de jogos, considera que esta é uma profissão com futuro?
    a. □ Sim.
    b. □ Não.
    c. □ Não aplicável.
10. Sabendo que a programação de jogos utiliza as linguagens de programação, considera adequada a utilização de uma aprendizagem baseada na programação de jogos para uma disciplina de programação?
    a. □ Sim.
    b. □ Não.
    c. □ Não aplicável.

(continue na página seguinte, por favor)

11. Sabendo que a programação de jogos necessita de metodologias avançadas de investigação e desenvolvimento poderá ser usada para disciplinas de engenharia de software ou analise de sistemas?
    a. □ Sim.
    b. □ Não.
    c. □ Não aplicável.

12. Em que nível de educação considera que a programação de jogos, pode ser inserida
    a. □ Secundário (10º – 12º anos de escolaridade)
    b. □ Licenciatura.
    c. □ Mestrado.
    d. □ Não sabe.

13. Que nível de importância atribui à necessidade de existirem cursos dedicados à programação de jogos

	Nada importante						**Muito importante**
13.1 Novos ramos de desenvolvimento e novas licenciaturas							
13.2 Criação de cursos superiores mais dedicados à programação de jogos							
13.3 Promover a formação na área							
13.4 Aplicar esta tecnologia noutros ambientes e processos de negócio							
13.5 Desenvolver novas características e competências para os estudantes							
13.6 Inovar produtos, procedimentos e serviços							

14. Quais são as linguagens de programação de jogos que conhece?
    14.1. □ Linguagem C
    14.2. □ Java
    14.3. □ Linguagem C++
    14.4. □ Unreal Engine
    14.5. □ Microsoft XNA
    14.6. □ OpenGL

(continue na página seguinte, por favor)

15. Quais são as maiores dificuldades para a definição de cursos superiores dedicados à programação de jogos?

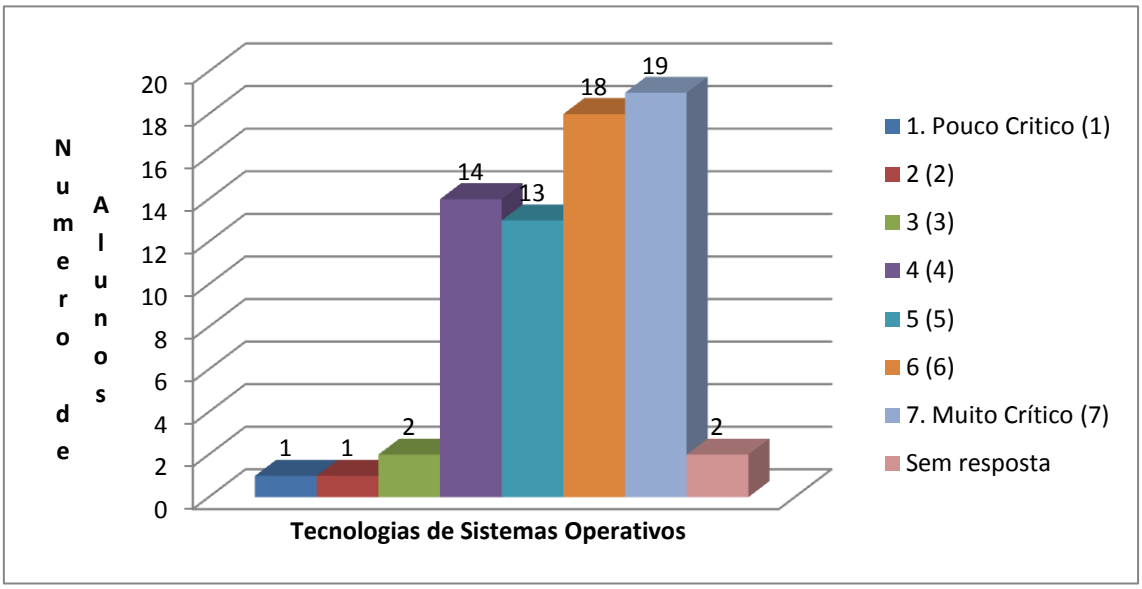
	Muito difícil						**Nada difícil**
15.1 Aspectos culturais							
15.2 Problemas de comunicação e exposição dos cursos							
15.3 Aceitação do publico em geral							
15.4 Carência de recursos humanos e competências na área							
15.5 Infra-estruturas tecnológicas inadequadas							
15.6 Carência de recursos financeiros							
15.7 Tempo disponibilizado pelas partes envolvidas							
15.8 Outros. (Por favor, especificar) - -							

Muito obrigada pela participação.

9.2. Resultados do Questionário

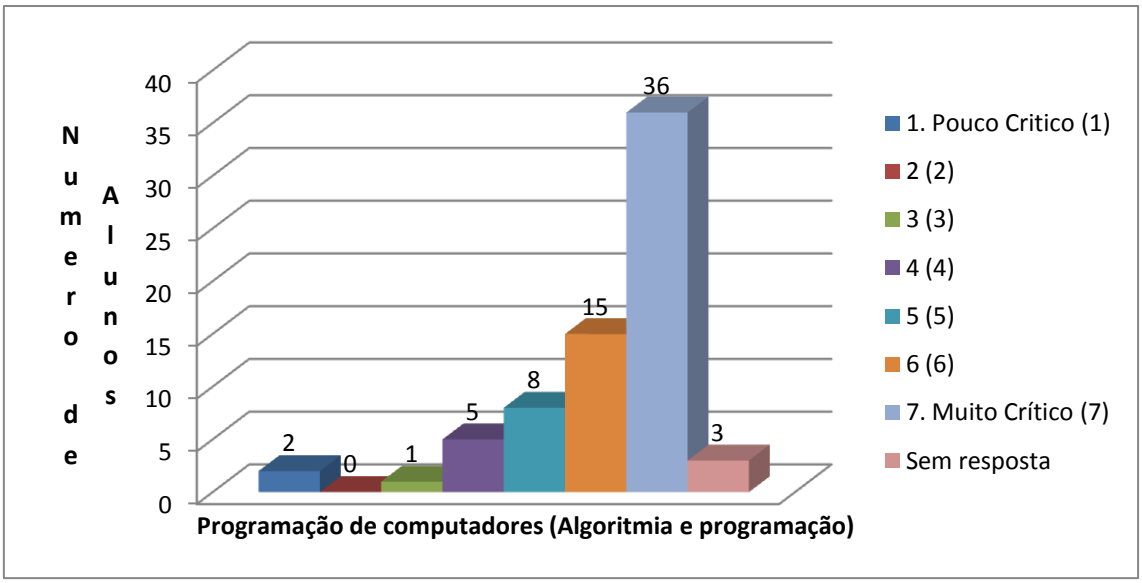
1. Que tipo de conhecimento considera critico para um Engenheiro Informático?
   1.1 Tecnologias de Sistemas Operativos

Resposta	Contagem	Percentagem
1. Pouco Critico (1)	1	1.43%
2 (2)	1	1.43%
3 (3)	2	2.86%
4 (4)	14	20.00%
5 (5)	13	18.57%
6 (6)	18	25.71%
7. Muito Crítico (7)	19	27.14%
Sem resposta	2	2.86%



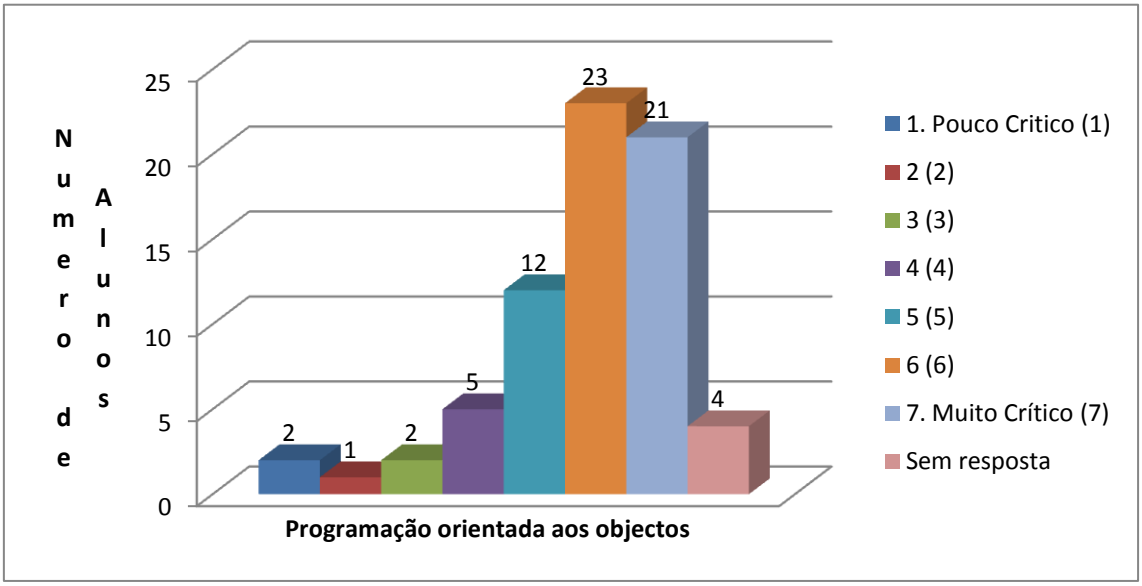
1.2 Programação de computadores (Algoritmia e programação)

Resposta	Contagem	Percentagem
1. Pouco Critico (1)	2	2.86%
2 (2)	0	0.00%
3 (3)	1	1.43%
4 (4)	5	7.14%
5 (5)	8	11.43%
6 (6)	15	21.43%
7. Muito Crítico (7)	36	51.43%
Sem resposta	3	4.29%



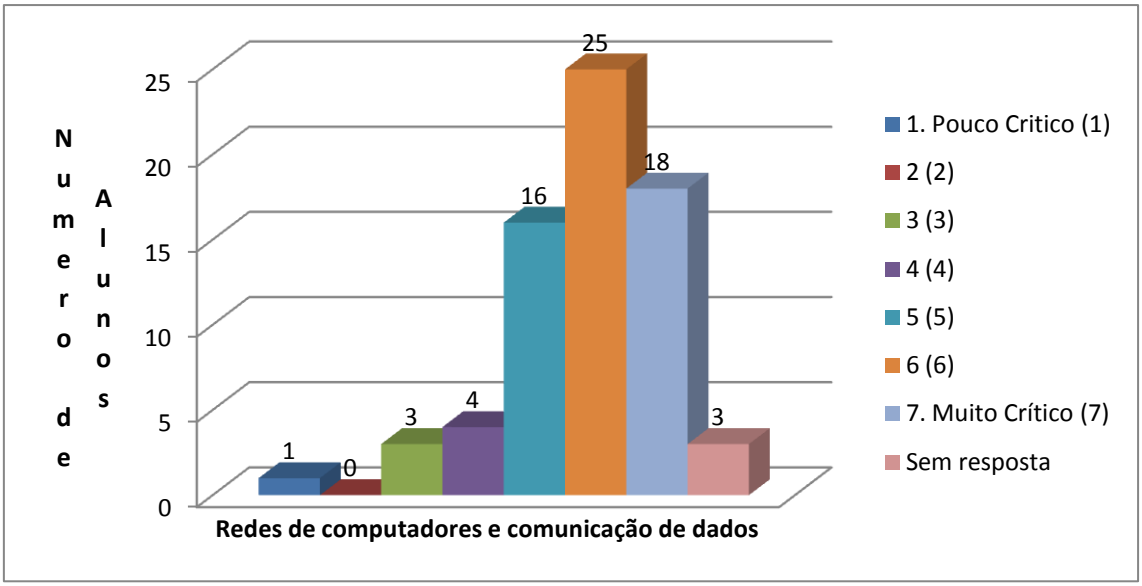
1.3 Programação orientada aos objectos

Resposta	Contagem	Percentagem
1. Pouco Critico (1)	2	2.86%
2 (2)	1	1.43%
3 (3)	2	2.86%
4 (4)	5	7.14%
5 (5)	12	17.14%
6 (6)	23	32.86%
7. Muito Crítico (7)	21	30.00%
Sem resposta	4	5.71%



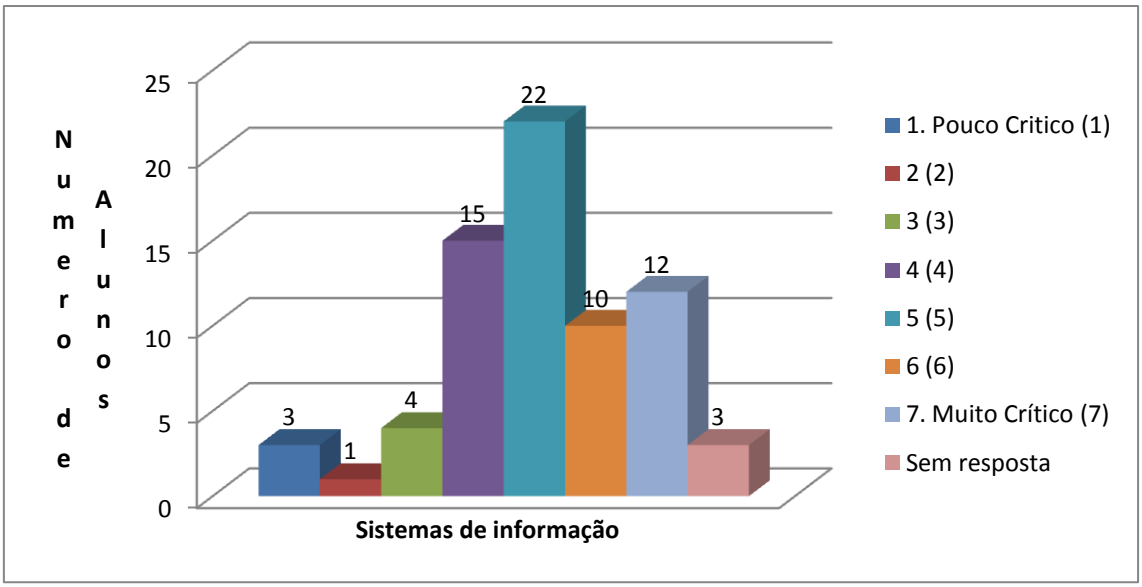
1.4 Redes de computadores e comunicação de dados

Resposta	Contagem	Percentagem
1. Pouco Critico (1)	1	1.43%
2 (2)	0	0.00%
3 (3)	3	4.29%
4 (4)	4	5.71%
5 (5)	16	22.86%
6 (6)	25	35.71%
7. Muito Crítico (7)	18	25.71%
Sem resposta	3	4.29%



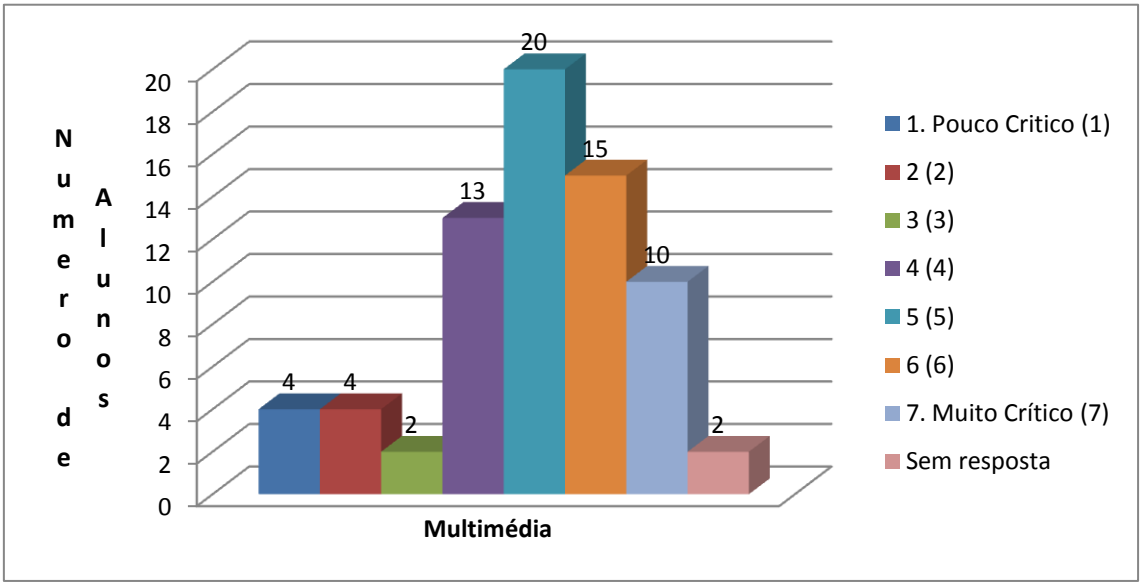
1.5 Sistemas de informação

Resposta	Contagem	Percentagem
1. Pouco Critico (1)	3	4.29%
2 (2)	1	1.43%
3 (3)	4	5.71%
4 (4)	15	21.43%
5 (5)	22	31.43%
6 (6)	10	14.29%
7. Muito Crítico (7)	12	17.14%
Sem resposta	3	4.29%



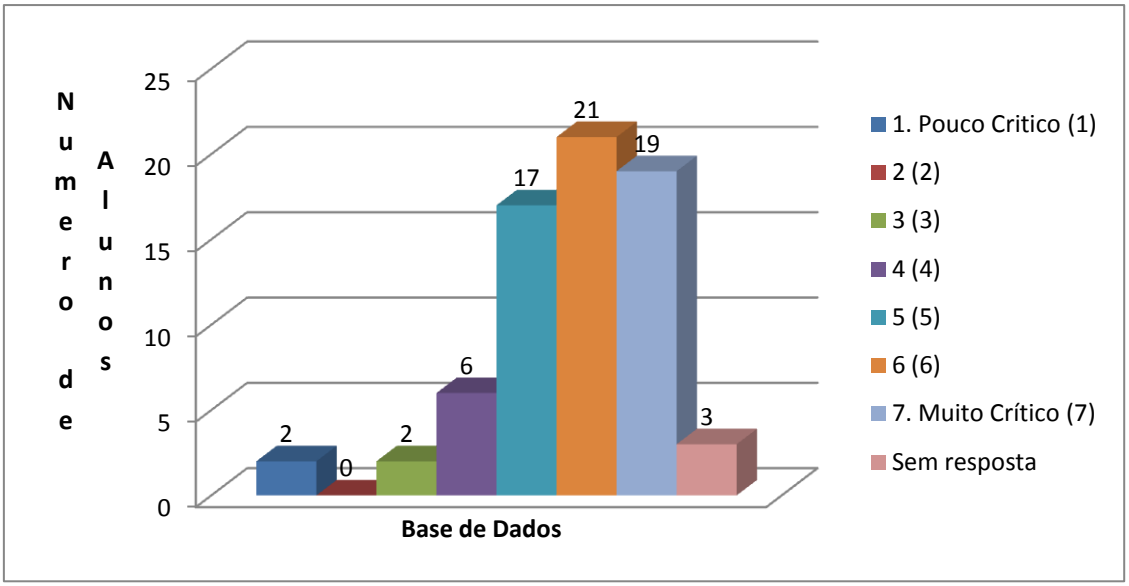
1.6 Multimédia

Resposta	Contagem	Percentagem
1. Pouco Critico (1)	4	5.71%
2 (2)	4	5.71%
3 (3)	2	2.86%
4 (4)	13	18.57%
5 (5)	20	28.57%
6 (6)	15	21.43%
7. Muito Crítico (7)	10	14.29%
Sem resposta	2	2.86%



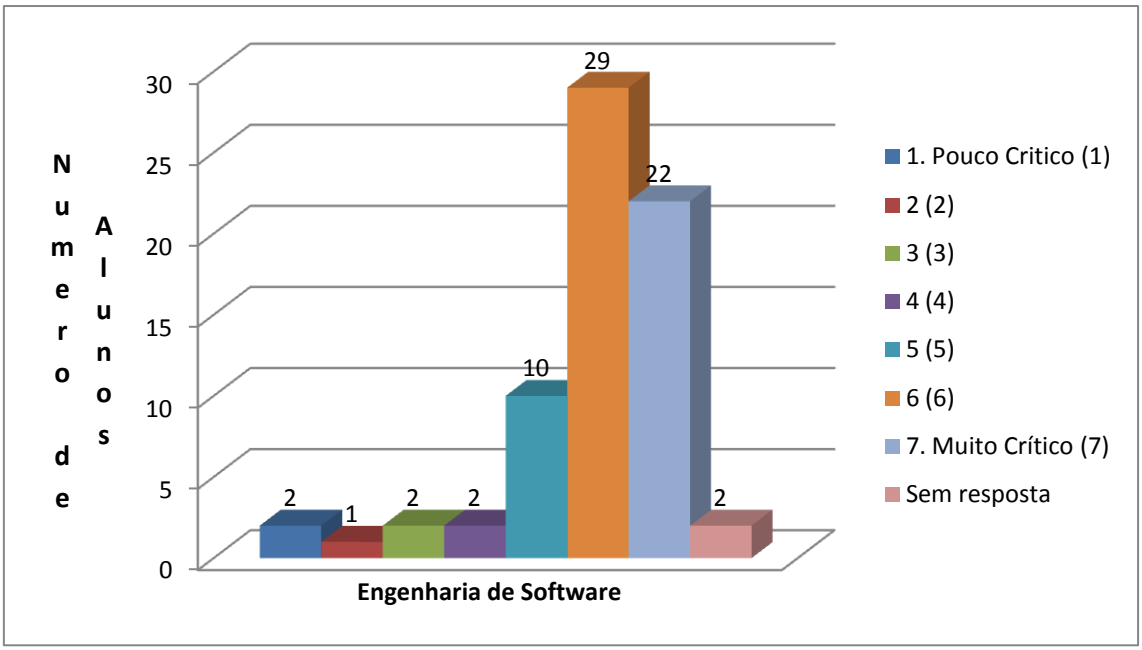
1.7 Base de Dados

Resposta	Contagem	Percentagem
1. Pouco Critico (1)	2	2.86%
2 (2)	0	0.00%
3 (3)	2	2.86%
4 (4)	6	8.57%
5 (5)	17	24.29%
6 (6)	21	30.00%
7. Muito Crítico (7)	19	27.14%
Sem resposta	3	4.29%



1.8 Engenharia de Software

Resposta	Contagem	Percentagem
1. Pouco Critico (1)	2	2.86%
2 (2)	1	1.43%
3 (3)	2	2.86%
4 (4)	2	2.86%
5 (5)	10	14.29%
6 (6)	29	41.43%
7. Muito Crítico (7)	22	31.43%
Sem resposta	2	2.86%

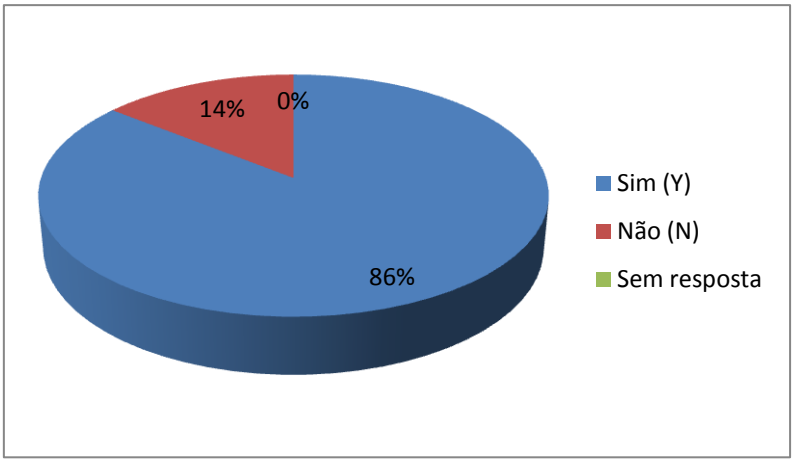


1.9 Outros. (Por favor, especificar)

Resposta	Contagem	Percentagem
Resposta	4	5.71%
Sem resposta	66	94.29%

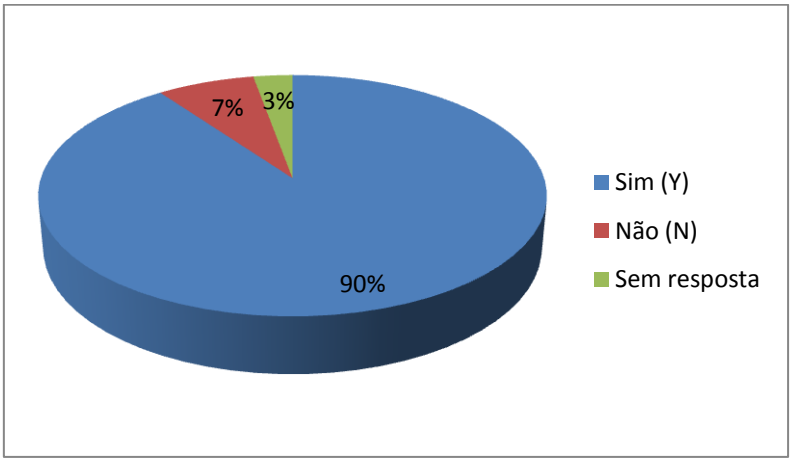
2. Gosta de jogos de computador?

Resposta	Contagem	Percentagem
Sim (Y)	60	85.71%
Não (N)	10	14.29%
Sem resposta	0	0.00%



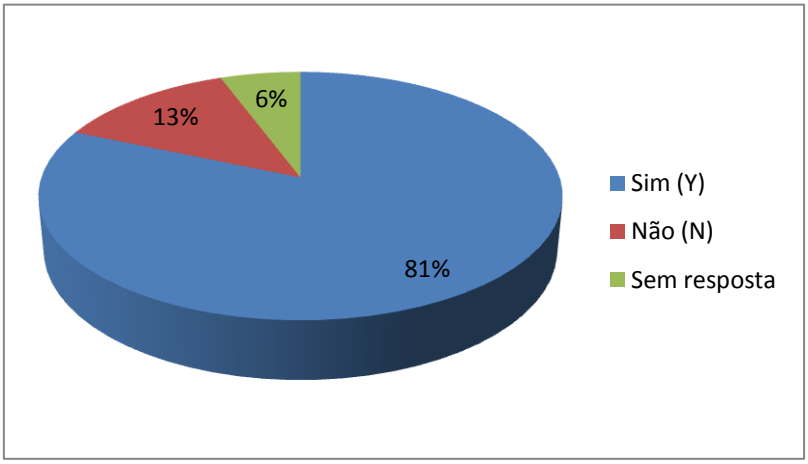
3. Gostava de saber como funcionam (e são programados) os jogos?

Resposta	Contagem	Percentagem
Sim (Y)	63	90.00%
Não (N)	5	7.14%
Sem resposta	2	2.86%



4. Gostaria de saber criar os seus próprios jogos?

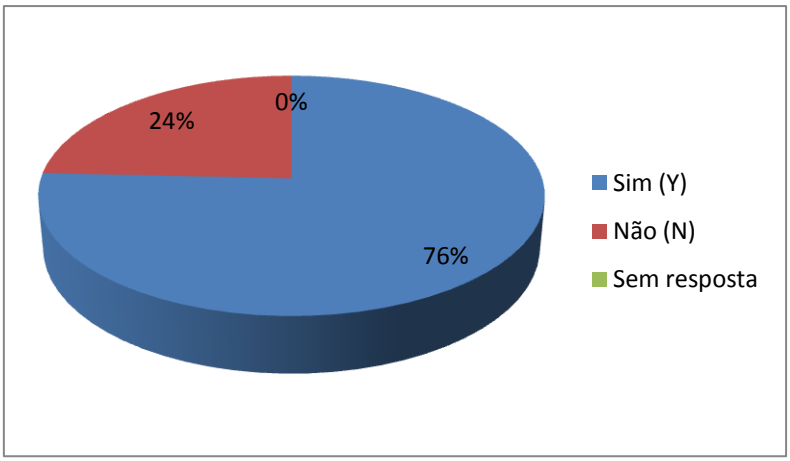
Resposta	Contagem	Percentagem
Sim (Y)	57	81.43%
Não (N)	9	12.86%
Sem resposta	4	5.71%



5. Considera que os jogos de computador são um factor de motivação para o ingresso de alunos no ensino superior no ramo da informática?

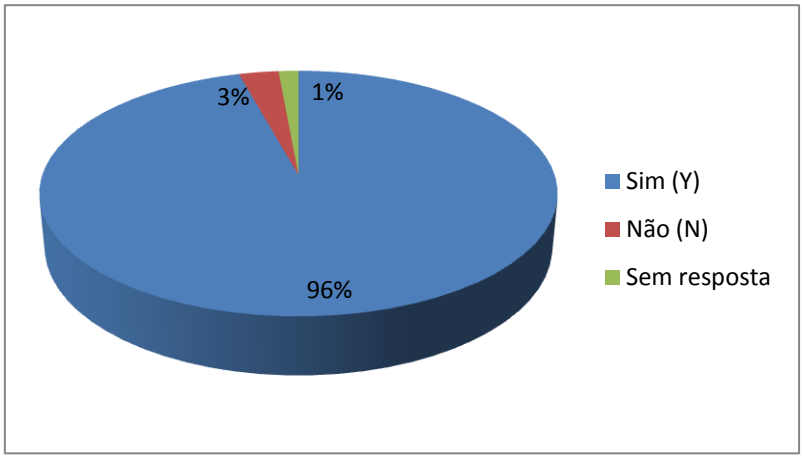
Resposta	Contagem	Percentagem

Sim (Y)	53	75.71%
Não (N)	17	24.29%
Sem resposta	0	0.00%



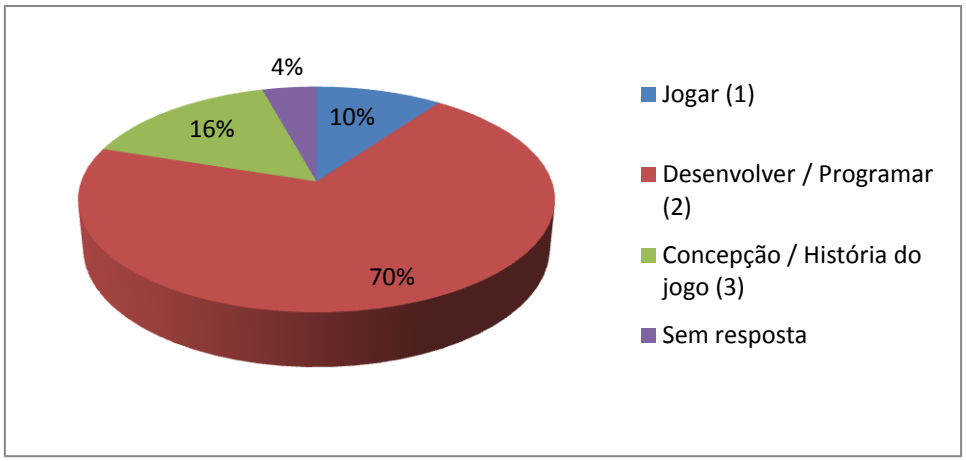
6. Acredita que a motivação é um factor essencial para o progresso do aluno no ensino superior?

Resposta	Contagem	Percentagem
Sim (Y)	67	95.71%
Não (N)	2	2.86%
Sem resposta	1	1.43%



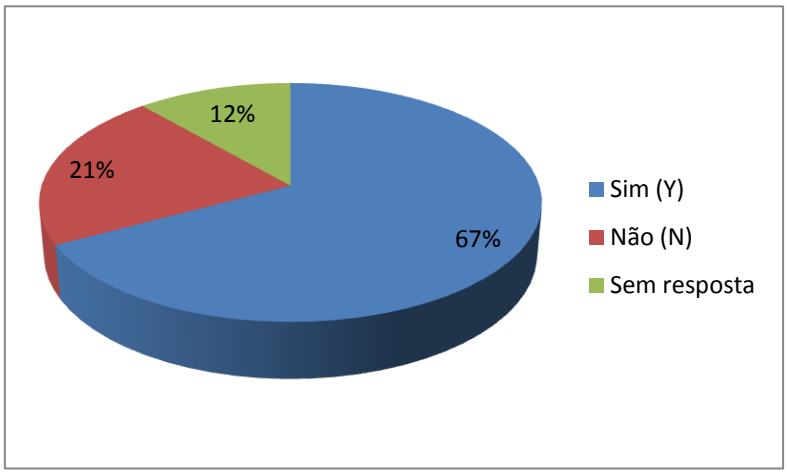
7. O que poderá motivar mais o progresso de um aluno?

Resposta	Contagem	Percentagem
Jogar (1)	7	10.00%
Desenvolver / Programar (2)	49	70.00%
Concepção / História do jogo (3)	11	15.71%
Sem resposta	3	4.29%



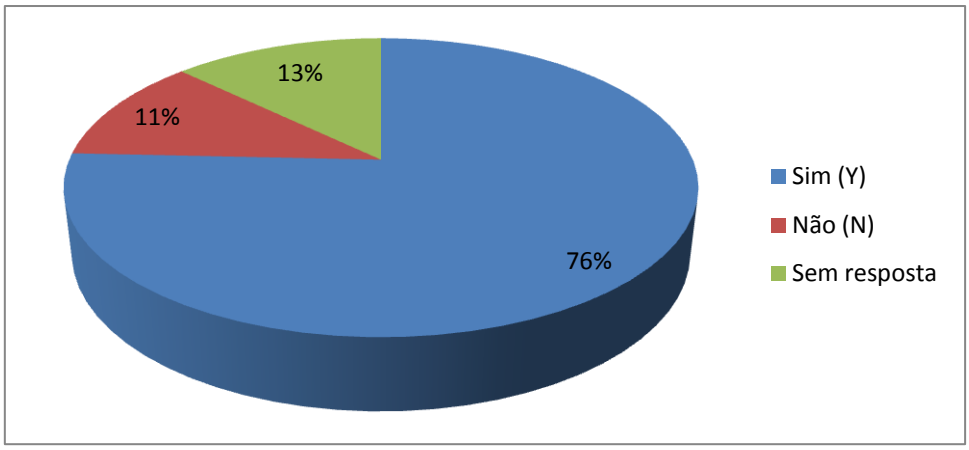
8. Concorda com uma lógica de aprendizagem baseada na programação de jogos de computador no ensino superior?

Resposta	Contagem	Percentagem
Sim (Y)	47	67.14%
Não (N)	15	21.43%
Sem resposta	8	11.43%



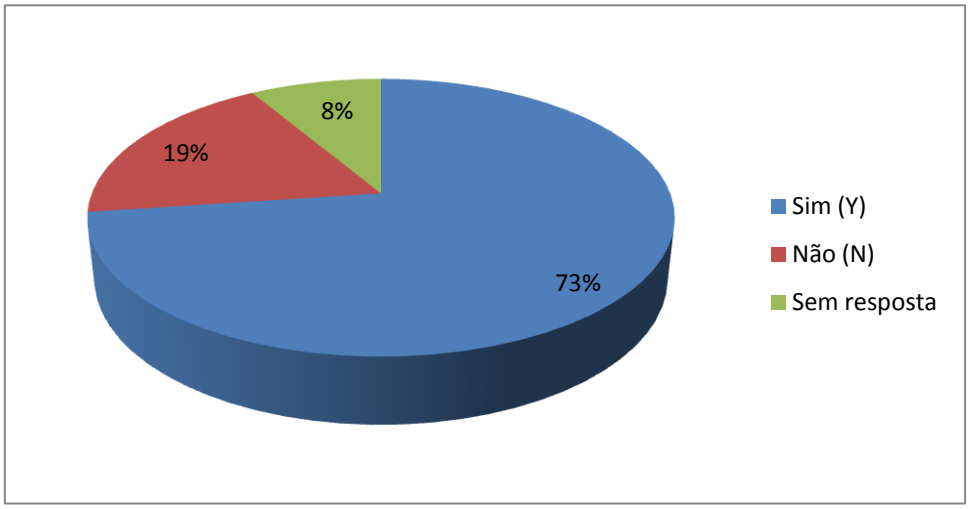
9. Uma vez feita a opção profissional, de programador de jogos, considera que esta é uma profissão com futuro?

Resposta	Contagem	Percentagem
Sim (Y)	53	75.71%
Não (N)	8	11.43%
Sem resposta	9	12.86%



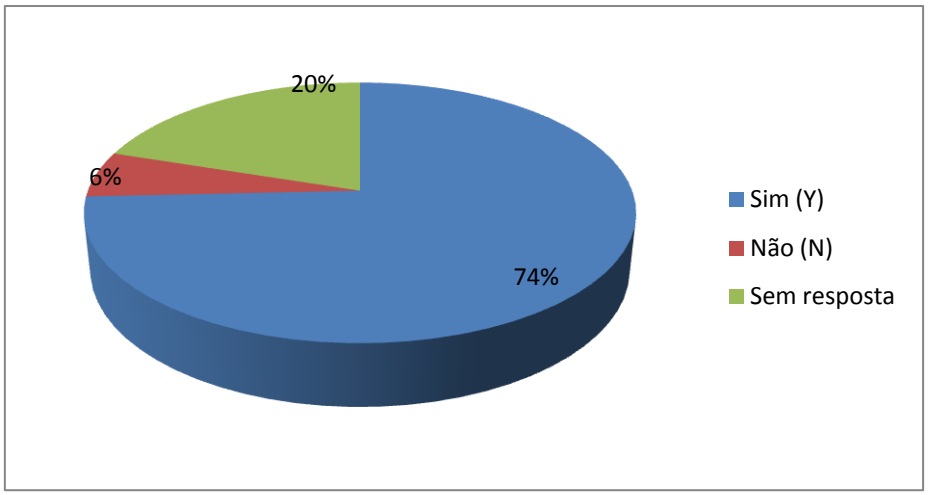
10. Sabendo que a programação de jogos utiliza as linguagens de programação, considera adequada a utilização de uma aprendizagem baseada na programação de jogos para uma disciplina de programação?

Resposta	Contagem	Percentagem
Sim (Y)	51	72.86%
Não (N)	13	18.57%
Sem resposta	6	8.57%



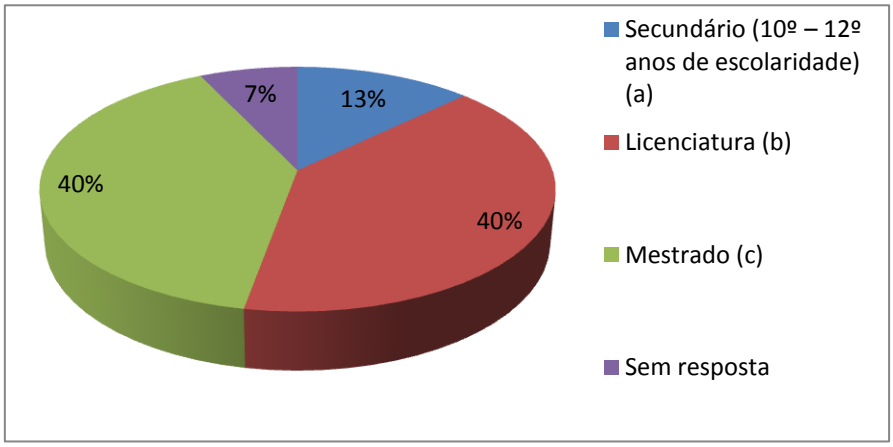
11. Sabendo que a programação de jogos necessita de metodologias avançadas de investigação e desenvolvimento poderá ser usada para disciplinas de engenharia de software ou analise de sistemas?

Resposta	Contagem	Percentagem
Sim (Y)	52	74.29%
Não (N)	4	5.71%
Sem resposta	14	20.00%



12. Em que nível de educação considera que a programação de jogos, pode ser inserida

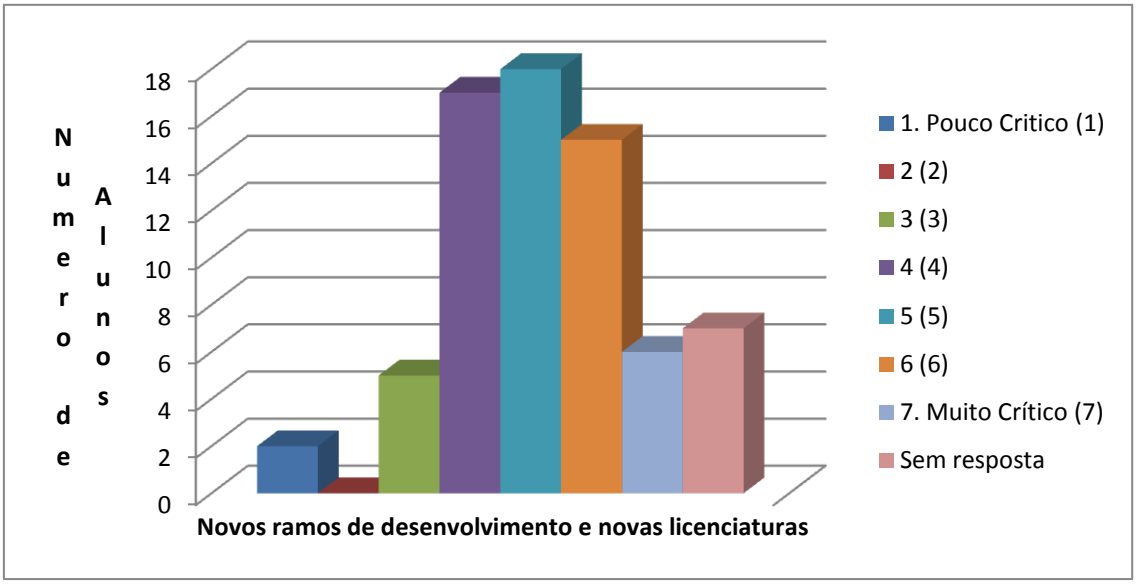
Resposta	Contagem	Percentagem
Secundário (10º – 12º anos de escolaridade) (a)	9	12.86%
Licenciatura (b)	28	40.00%
Mestrado (c)	28	40.00%
Sem resposta	5	7.14%



13. Que nível de importância atribui à necessidade de existirem cursos dedicados à programação de jogos

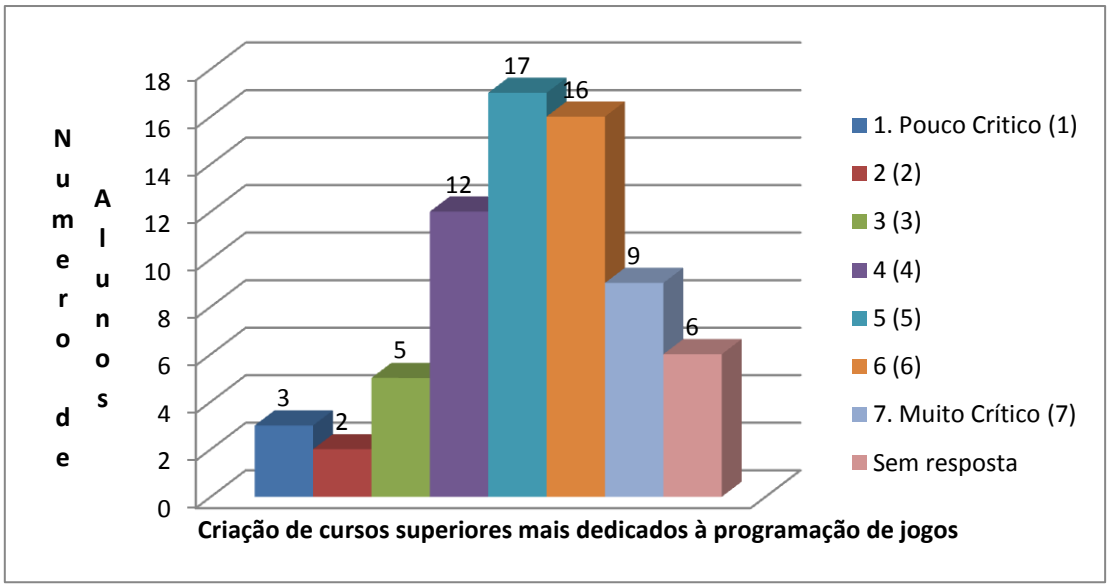
13.1 Novos ramos de desenvolvimento e novas licenciaturas

Resposta	Contagem	Percentagem
1. Pouco Critico (1)	2	2.86%
2 (2)	0	0.00%
3 (3)	5	7.14%
4 (4)	17	24.29%
5 (5)	18	25.71%
6 (6)	15	21.43%
7. Muito Crítico (7)	6	8.57%
Sem resposta	7	10.00%



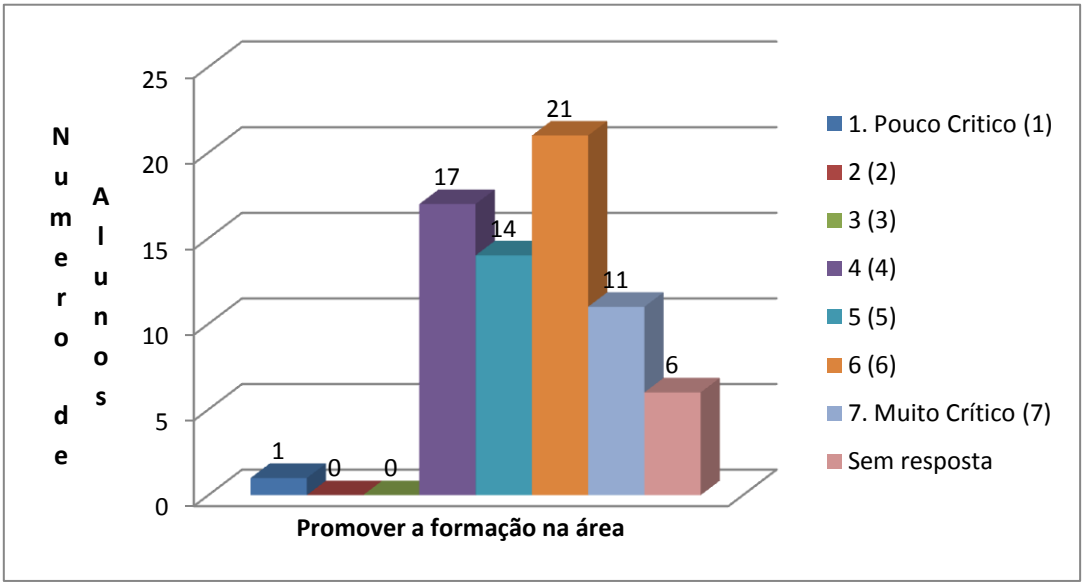
13.2 Criação de cursos superiores mais dedicados à programação de jogos

Resposta	Contagem	Percentagem
1. Pouco Critico (1)	3	4.29%
2 (2)	2	2.86%
3 (3)	5	7.14%
4 (4)	12	17.14%
5 (5)	17	24.29%
6 (6)	16	22.86%
7. Muito Crítico (7)	9	12.86%
Sem resposta	6	8.57%



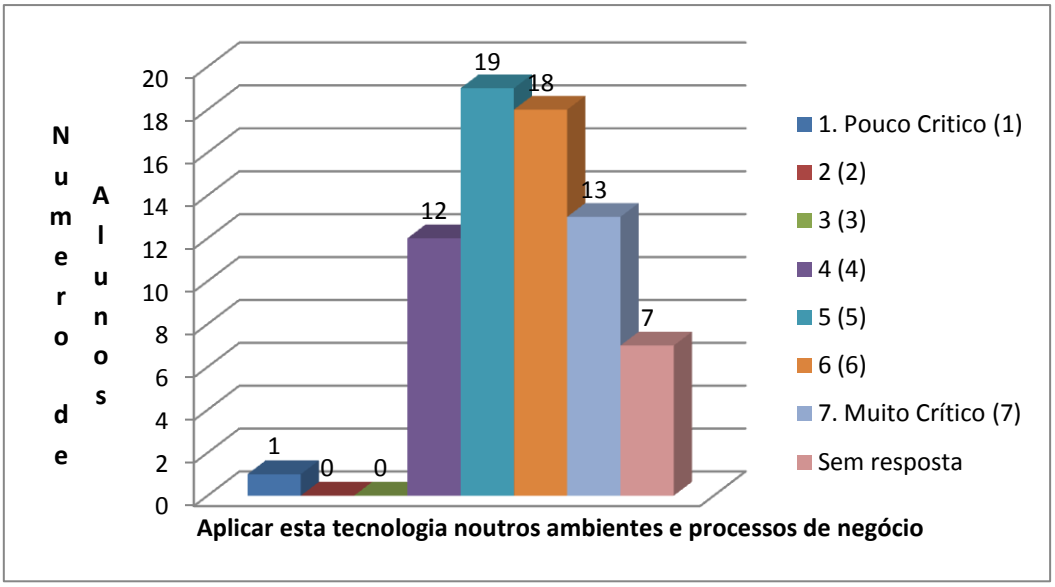
13.3 Promover a formação na área

Resposta	Contagem	Percentagem
1. Pouco Critico (1)	1	1.43%
2 (2)	0	0.00%
3 (3)	0	0.00%
4 (4)	17	24.29%
5 (5)	14	20.00%
6 (6)	21	30.00%
7. Muito Crítico (7)	11	15.71%
Sem resposta	6	8.57%



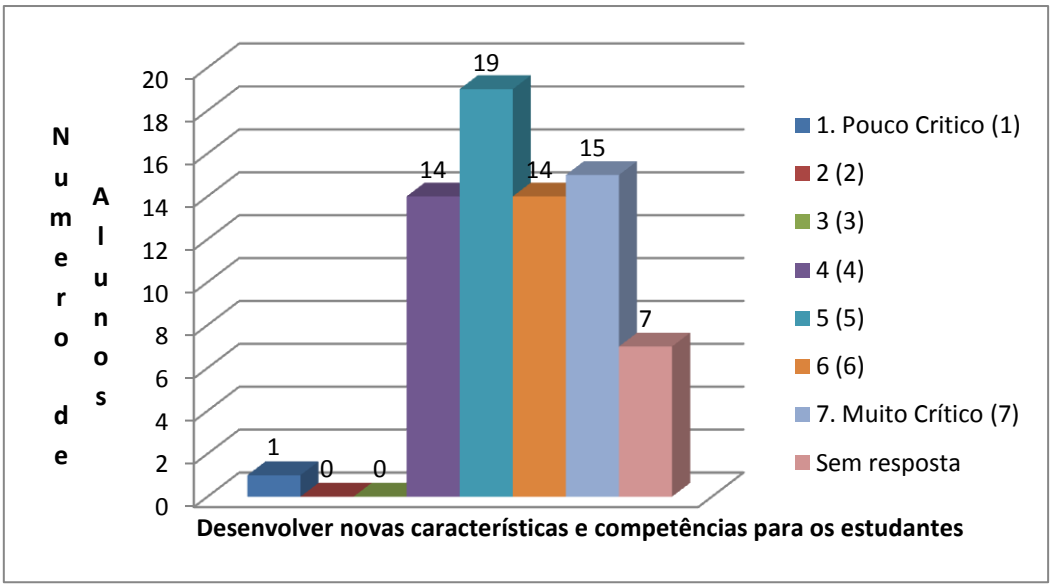
13.4 Aplicar esta tecnologia noutros ambientes e processos de negócio

Resposta	Contagem	Percentagem
1. Pouco Critico (1)	1	1.43%
2 (2)	0	0.00%
3 (3)	0	0.00%
4 (4)	12	17.14%
5 (5)	19	27.14%
6 (6)	18	25.71%
7. Muito Crítico (7)	13	18.57%
Sem resposta	7	10.00%



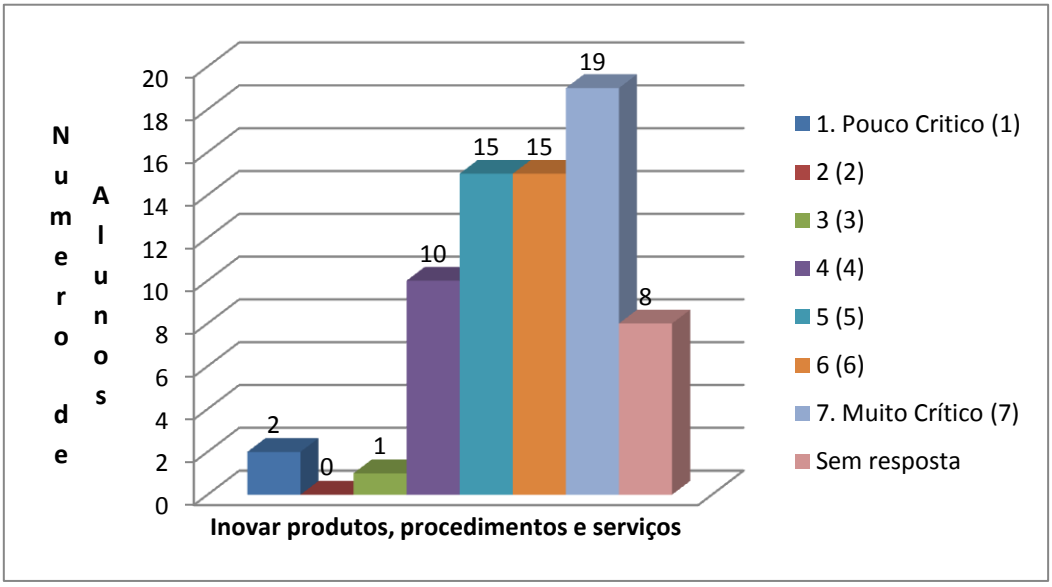
13.5 Desenvolver novas características e competências para os estudantes

Resposta	Contagem	Percentagem
1. Pouco Critico (1)	1	1.43%
2 (2)	0	0.00%
3 (3)	0	0.00%
4 (4)	14	20.00%
5 (5)	19	27.14%
6 (6)	14	20.00%
7. Muito Crítico (7)	15	21.43%
Sem resposta	7	10.00%



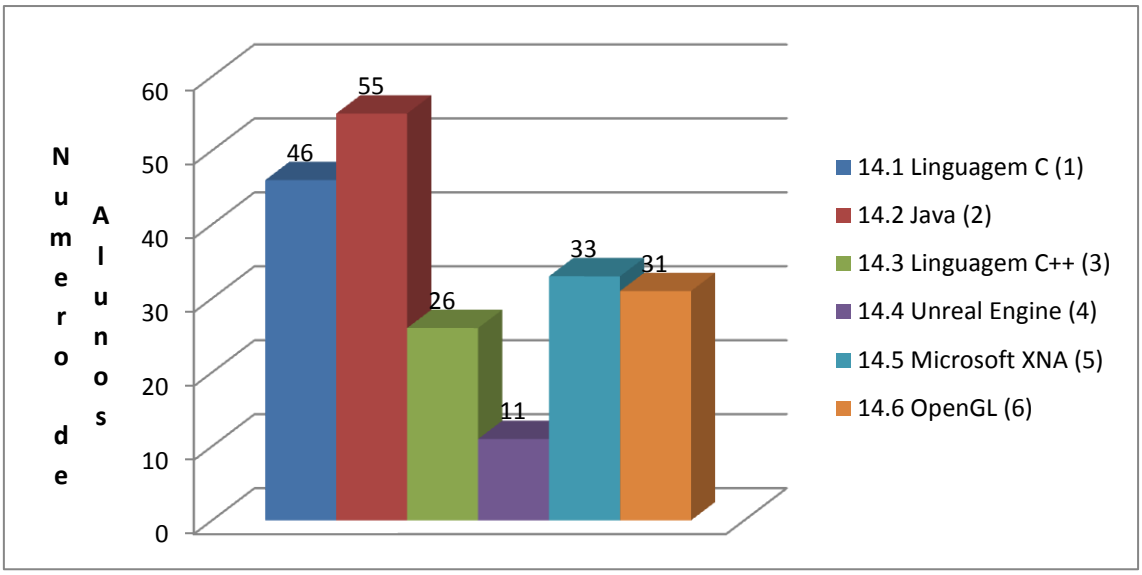
13.6 Inovar produtos, procedimentos e serviços

Resposta	Contagem	Percentagem
1. Pouco Critico (1)	2	2.86%
2 (2)	0	0.00%
3 (3)	1	1.43%
4 (4)	10	14.29%
5 (5)	15	21.43%
6 (6)	15	21.43%
7. Muito Crítico (7)	19	27.14%
Sem resposta	8	11.43%



14. Quais são as linguagens de programação de jogos que conhece?

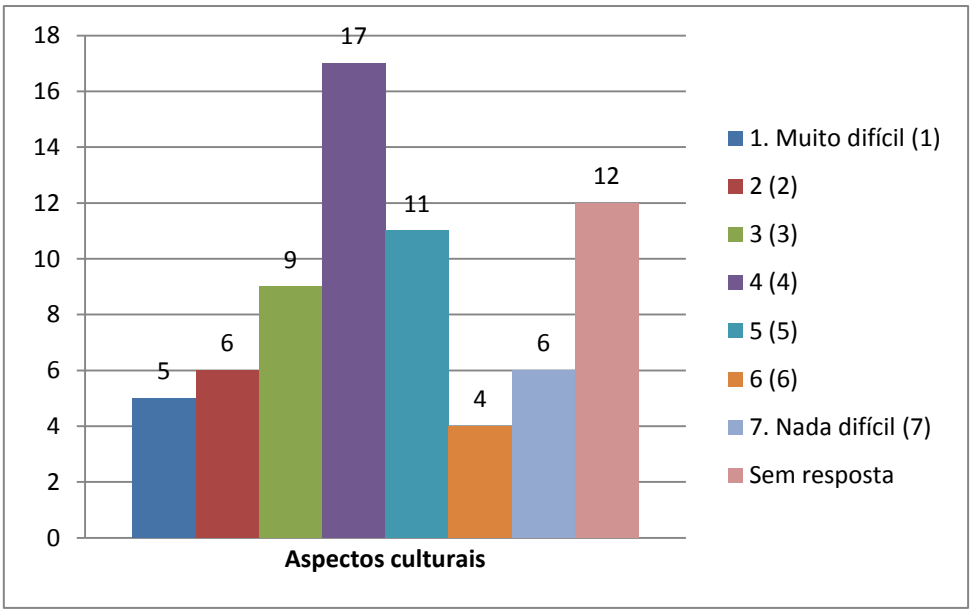
Resposta	Contagem	Percentagem
14.1 Linguagem C (1)	46	65.71%
14.2 Java (2)	55	78.57%
14.3 Linguagem C++ (3)	26	37.14%
14.4 Unreal Engine (4)	11	15.71%
14.5 Microsoft XNA (5)	33	47.14%
14.6 OpenGL (6)	31	44.29%



15. Quais são as maiores dificuldades para a definição de cursos superiores dedicados à programação de jogos?

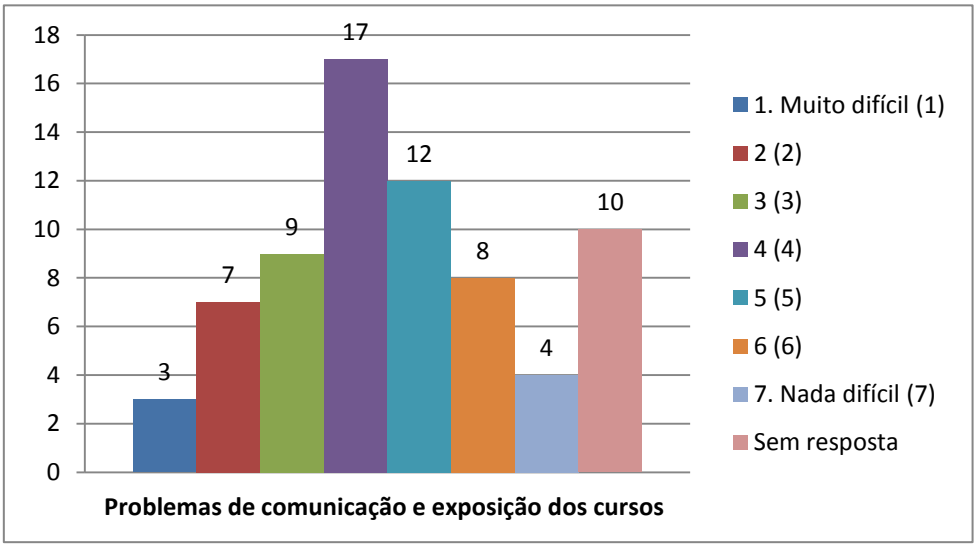
15.1 Aspectos culturais

Resposta	Contagem	Percentagem
1. Muito difícil (1)	5	7.14%
2 (2)	6	8.57%
3 (3)	9	12.86%
4 (4)	17	24.29%
5 (5)	11	15.71%
6 (6)	4	5.71%
7. Nada difícil (7)	6	8.57%
Sem resposta	12	17.14%



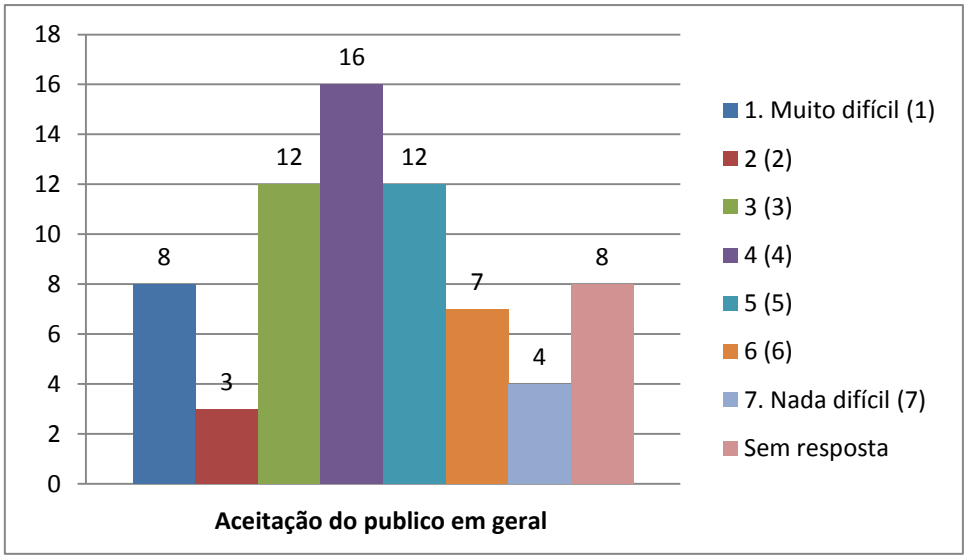
15.2 Problemas de comunicação e exposição dos cursos

Resposta	Contagem	Percentagem
1. Muito difícil (1)	3	4.29%
2 (2)	7	10.00%
3 (3)	9	12.86%
4 (4)	17	24.29%
5 (5)	12	17.14%
6 (6)	8	11.43%
7. Nada difícil (7)	4	5.71%
Sem resposta	10	14.29%



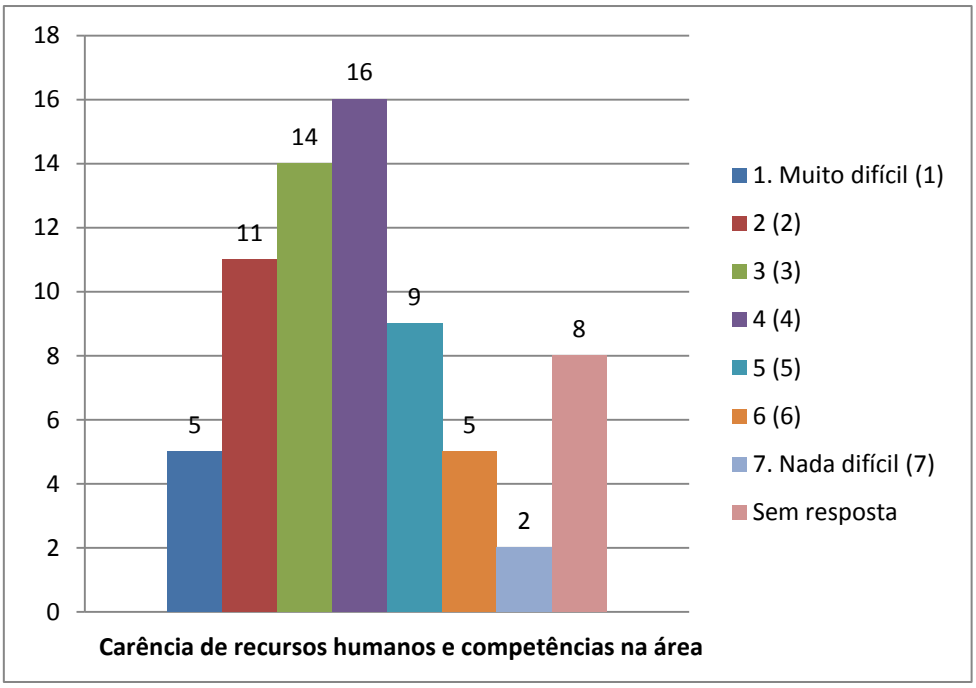
15.3 Aceitação do publico em geral

Resposta	Contagem	Percentagem
1. Muito difícil (1)	8	11.43%
2 (2)	3	4.29%
3 (3)	12	17.14%
4 (4)	16	22.86%
5 (5)	12	17.14%
6 (6)	7	10.00%
7. Nada difícil (7)	4	5.71%
Sem resposta	8	11.43%



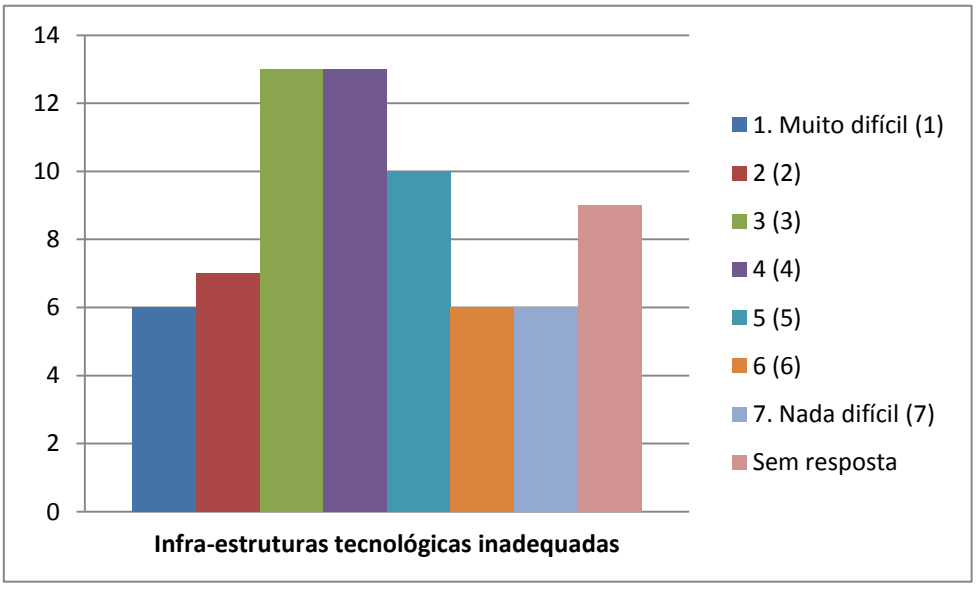
15.4 Carência de recursos humanos e competências na área

Resposta	Contagem	Percentagem
1. Muito difícil (1)	5	7.14%
2 (2)	11	15.71%
3 (3)	14	20.00%
4 (4)	16	22.86%
5 (5)	9	12.86%
6 (6)	5	7.14%
7. Nada difícil (7)	2	2.86%
Sem resposta	8	11.43%



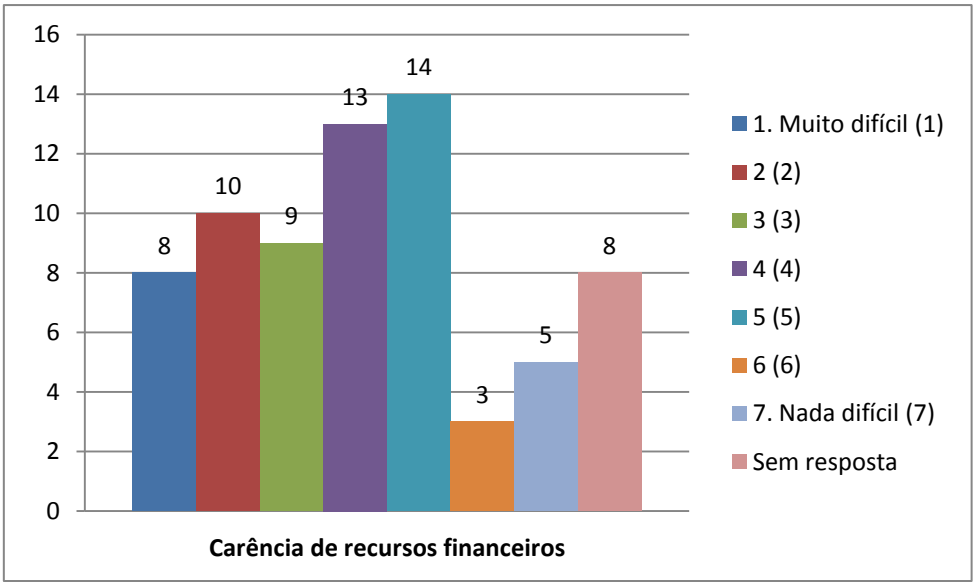
15.5 Infra-estruturas tecnológicas inadequadas

Resposta	Contagem	Percentagem
1. Muito difícil (1)	6	8.57%
2 (2)	7	10.00%
3 (3)	13	18.57%
4 (4)	13	18.57%
5 (5)	10	14.29%
6 (6)	6	8.57%
7. Nada difícil (7)	6	8.57%
Sem resposta	9	12.86%



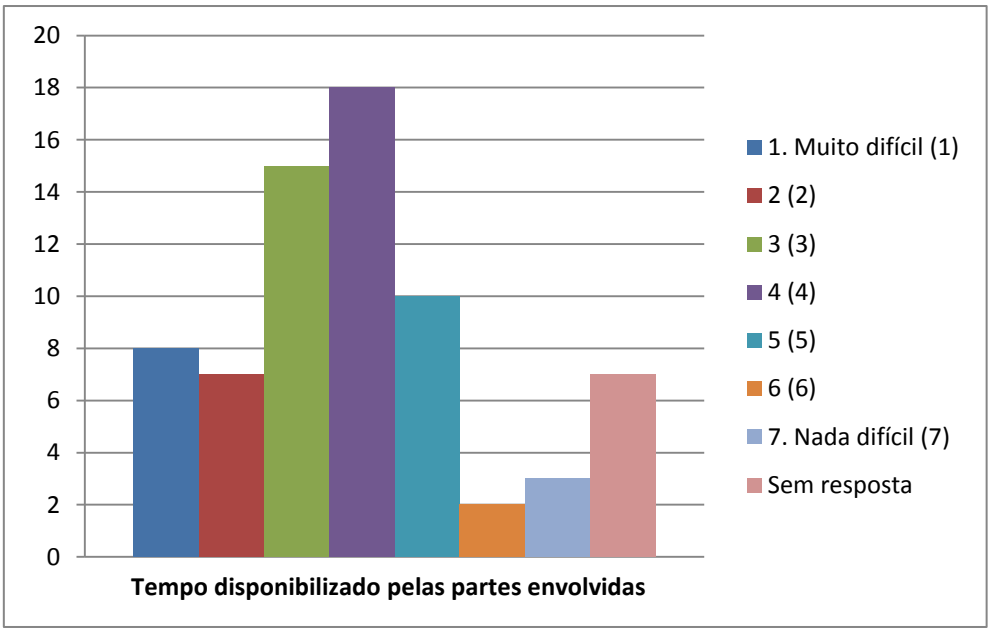
15.6 Carência de recursos financeiros

Resposta	Contagem	Percentagem
1. Muito difícil (1)	8	11.43%
2 (2)	10	14.29%
3 (3)	9	12.86%
4 (4)	13	18.57%
5 (5)	14	20.00%
6 (6)	3	4.29%
7. Nada difícil (7)	5	7.14%
Sem resposta	8	11.43%



15.7 Tempo disponibilizado pelas partes envolvidas

Resposta	Contagem	Percentagem
1. Muito difícil (1)	8	11.43%
2 (2)	7	10.00%
3 (3)	15	21.43%
4 (4)	18	25.71%
5 (5)	10	14.29%
6 (6)	2	2.86%
7. Nada difícil (7)	3	4.29%
Sem resposta	7	10.00%



15.8 Outros. (Por favor, especificar)

Resposta	Contagem	Percentagem
Resposta	2	2.86%
Sem resposta	68	97.14%