FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Jogo Sério Colaborativo para o Ensino da Programação a Crianças

Admilo Ribeiro

Mestrado Integrado em Engenharia Informática e Computação

Orientador: António Coelho (Doutor)

Co-orientador: Ademar Aguiar (Doutor)

Fevereiro de 2012

Jogo Sério Colaborativo para o Ensino da Programação a Crianças

Admilo Ribeiro

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo júri:

Presidente: Rui Camacho (Doutor)

Vogal Externo: Ramires Fernandes (Doutor)

Orientador: António Coelho (Doutor)

Fevereiro de 2012

Resumo

As crianças têm uma relação muito íntima com o ato de brincar. Brincam para experimentar coisas, conhecer o seu meio ambiente, divertirem-se e sobretudo para se relacionarem uns com os outros.

O aparecimento dos computadores pessoais trouxe também novas formas de brincar - os videojogos. Mas brincar não é só divertimento, é também um processo de aprendizagem e como tal os videojogos tornam-se num ótimo meio para aprendizagem, aproveitando o seu grande potencial motivacional.

Por outro lado, a programação é ainda vista como uma atividade técnica, restrita a um pequeno segmento da população. Este pensamento deve-se ao facto, de a aprendizagem dos conceitos da programação apresentarem um conjunto de barreiras iniciais, como a necessidade de conhecer a sintaxe e os comandos da linguagens, aprender a solucionar os problemas de forma estruturada e aprender como os programas são executados. Estas barreiras são ainda maiores quando estamos a falar de crianças.

Desta forma, esta dissertação é sobre projetar, criar e testar um jogo sério para introduzir conceitos de programação às crianças - Boobo World.

O projeto do Boobo World contém duas partes: a primeira que descreve a história do jogo, a sua mecânica, arte e software, e uma segunda parte que diz respeito à componente pedagógica a ser transmitida. Foi também desenvolvido um estudo do estado da arte sobre plataformas de jogos, jogos e ambientes de programação para crianças, já desenvolvidos, determinando as suas limitações e analisando os pontos fortes de forma a conceber um novo sistema de ensino dos conceitos da programação de forma fácil, colaborativa e divertida. Também nesta fase desenvolveu-se um processo criativo, com o objetivo de criar algo diferente do que já existe.

A criação consistiu na implementação de um protótipo como prova de conceito do trabalho desenvolvido na primeira parte. O protótipo contém as principais ideias e conceitos do jogo desenhado, permitindo assim uma validação das mesmas junto das crianças.

Por fim, a fase de teste consistiu na apresentação do protótipo desenvolvido, juntamente com outra solução(Scratch), a um grupo de crianças. Este teste teve como objetivo determinar as preferências de um jogo em relação ao outro, e determinar as reações das crianças a uma caraterística muito particular do Boobo World - a colaboração.

Abstract

Children have a very intimate relationship with the act of playing, experiencing and exploring the environment and, establishing relationships between each other.

The invention of the personal computer brought new ways to play - the video games. But playing is not only for fun, it's also a learning process, so videogames can be used as a way to improve the learning process, taking advantage of its great motivational potential.

On the other hand, programming is widely seen as a technical and restricted activity that is only accessible to a small group of people. This reasoning is derived by the problem regarding the initial barriers presented to those that want to learn programming. These include the need to know the syntax and the commands of the languages, to learn how to find and structure the solution and understand how programs are executed. These barriers are even greater when we talk about teaching programming to children.

So this dissertation presents the game design, development and testing of serious game to teach programming to children - Boobo World.

The design of Boobo World contains two parts: the first is related to the game development, presenting the story, mechanics, concept art and development, and a second part which relates to the educational components to be transmitted. A study has also been conducted on the state of the art in game platforms, games and programming environments for children, exposing their limitations and strengths, in order to assist the design of this new game to teach programming concepts easily, collaborative and in a funny way.

The development phase consisted in the implementation of a prototype as a proof of concept for the work, the main ideas and concepts of the game designed, allowing a validation with children.

Finally, the test phase consisted in comparing the prototype, with another solution(Scratch), for a group of children. This test aimed to determine the preferences by children of one system over another and determine their reactions to a very particular feature of Boobo World - collaboration.

Conteúdo

1 Introdução **1**
- 1.1 Enquadramento . . . 1
- 1.2 A programação e os problemas do seu ensino a crianças . . . 2
- 1.3 Objetivos . . . 3
- 1.4 Estrutura do Documento . . . 3

2 Jogos de Computador **5**
- 2.1 Géneros . . . 6
- 2.2 Serious Games . . . 7
- 2.3 Componente Multiplayer e Colaboração . . . 11
- 2.4 Desenho de um Jogo . . . 15
  - 2.4.1 Ínicio . . . 15
  - 2.4.2 O primeiro Documento . . . 15
  - 2.4.3 Ferramentas . . . 17
  - 2.4.4 Riscos . . . 17
  - 2.4.5 Lançamento do Jogo . . . 17
- 2.5 Resumo . . . 18

3 Linguagens e Ambientes de Programação para Crianças **19**
- 3.1 Linguagens e Sistemas de Ensino da Programação . . . 20
  - 3.1.1 Ensinar a Criar . . . 21
  - 3.1.2 Ensinar a Estruturar . . . 26
  - 3.1.3 Compreender a Execução . . . 27
  - 3.1.4 Suporte ao Ensino . . . 29
- 3.2 Programação como Suporte/Capacitação para outras áreas . . . 30
  - 3.2.1 Logo . . . 31
  - 3.2.2 AgentSheets . . . 32
  - 3.2.3 Squeak Etoys . . . 33
- 3.3 Resumo . . . 34

4 Boobo World - Game Design **35**
- 4.1 Conceito . . . 35
- 4.2 Objetivos . . . 35
- 4.3 Linguagem e Ambiente Programação . . . 36
- 4.4 Mecânica . . . 39
- 4.5 Níveis . . . 40
- 4.6 Avatar e Robô . . . 42
- 4.7 Interface do Utilizador . . . 43

4.8 Controlos 44
4.9 Arquitetura 45
4.10 Tecnologias e Ferramentas 46

5 Boobo World - Implementação **49**
5.1 Mundo Virtual 49
5.2 Níveis 50
5.3 Desafios e Colaboração 52
5.4 Instruções e Ambiente de programação 55

6 Boobo World - Resultados e Avaliação **57**
6.1 Desenho do teste 57
6.2 Realização do teste 58
6.3 Análise dos Resultados 59
6.4 Críticas e Sugestões 61

7 Conclusões **63**
7.1 Satisfação dos Objetivos 64
7.2 Trabalho Futuro 64

A Materiais utilizados no teste **65**
A.1 Guião do Teste 65
A.2 Folha de Recolha de Resultados 67

Referências **71**

Lista de Figuras

2.1 Jogos de computador no mundo dos jogos 5
2.2 O *produtor* e jogador têm perspetivas diferentes 6
2.3 Diferença entre um *Serious Game* e um *Game* [QN08] 8
2.4 Taxonomia dos *Serious Games* [SS08]. 9
2.5 Interface Flight Simulator. 10
2.6 Interface Genomics Digital Lab. 11
2.7 Interface do Moshi Monsters. 12
2.8 Interface do Club Penguin. 13
2.9 Mundo virtual do Driving Kids 14
2.10 Um puzzle de memória no Driving Kids 14
2.11 Componentes de um documento de desenho do jogo. 15

3.1 Linguagens e Sistemas de Ensino dos mecanismos da Programação 20
3.2 Interface do Alice. 22
3.3 Interface do Greenfoot. 23
3.4 Interface do Scratch. 24
3.5 Kodu- Ambiente de programação e linguagem. 25
3.6 Kodu Game Lab - Acesso aos projetos partilhados. 25
3.7 Interface do Kara. 26
3.8 Interface do Liveworld. 27
3.9 Interface do Mundo de Karel. 28
3.10 Ambiente Toon Talk. 28
3.11 Interface do MOOSE Crossing. 29
3.12 As três metáforas utilizadas por cleogo[CB98]. 30
3.13 Interface Logo(Logo Turtle). 31
3.14 Interface do AgentSheets. 32
3.15 Interface do Etoys. 33

4.1 Esboço ambiente programação no Boobo World. 38
4.2 Diagrama atividades- Mecânica do Boobo World 39
4.3 Comportamento especial do Boobo no Mundo Virtual 42
4.4 Diagrama Casos de uso com o teclado. 44
4.5 Diagrama casos de uso com o rato. 44
4.6 Arquitetura cliente/servidor 45

5.1 Terreno do mundo. 50
5.2 Nível 1 - Ensinar o boobo a falar. 50
5.3 Nível 2 - Ensinar o boobo a parar. 51
5.4 Nível 3 - Ensinar o boobo a voltar ao ponto de partida. 51

5.5 Um desafio no Boobo World. 52
5.6 Convidar outros jogadores para resolver um desafio. 52
5.7 Diagrama Sequência processo colaboração no Boobo World. 54
5.8 Instruções I - Sensores e Filtros. 55
5.9 Instruções II - Ações. 55
5.10 Instruções III - Seletores e Modificadores. 56
5.11 Ambiente programação no Boobo World. 56

6.1 Participantes no teste. 59
6.2 Performance das Crianças na resolução dos desafios. 59
6.3 Scratch vs Boobo World - Preferências das crianças. 60
6.4 Opinião das Crianças em relação a Colaboração no Boobo World 60

A.1 Folha de recolha de resultados 1/3 67
A.2 Folha de recolha de resultados 2/3 68
A.3 Folha de recolha de resultados 3/3 69

Lista de Tabelas

2.1 Géneros dos jogos de computador/videojogos. 7

3.1 Alternativas a Digitação do Código 21
3.2 Abordagens dos sistemas de suporte 31

4.1 Desafios simples no Boobo World. 41
4.2 Informações presentes no Boobo World. 43

LISTA DE TABELAS

Abreviaturas

AST	Abstract Syntax Tree
MDA	Mechanics-Dynamics-Aesthetics
MMOG	Massively Multiplayer Online Games
MMORPG	Massively multiplayer online role-playing games
MUD	Multi-User Dungeon
RPG	Role-Playing Games
RTS	Real Time Strategy
TBS	Turned-Based Strategy
TIC	Tecnologias da Informação e Comunicação

Capítulo 1

Introdução

"A Brincar também se Aprende."

O ato de brincar é reconhecido pelos psicólogos e antropólogos como a chave para a formação do relacionamento entre as crianças e os seus corpos, a utilização de ferramentas, a envolvência na comunidade e a aquisição do conhecimento. É brincando que as crianças experimentam papéis e aprendem a manipular e explorar recursos no seu ambiente. À medida que vão crescendo, o ato de brincar pode motivar outras formas de aprendizagem [Jen09].

Com o desenvolvimento dos primeiros computadores pessoais, também surgiu uma nova atividade lúdica - o videojogo.

Os videojogos são atualmente o mercado mais rentável na indústria de entretenimento [myG11]. E este sucesso deve-se ao fato de os videojogos serem atividades bastante apelativas, levando pessoas, em especial as crianças e jovens, a passarem muitas das horas disponíveis nesta atividade. Porém, os videojogos não servem apenas para entreter, também são meios para o ensino e aprendizagem [Pre02].

1.1 Enquadramento

Este trabalho será realizado num ambiente misto académico-empresarial, entre a FEUP e a empresa Tecla Colorida, integrando-se com um produto da empresa, a plataforma escolinhas.pt.

Plataforma escolinhas.pt

Escolinhas.pt é uma plataforma colaborativa e social para escolas do ensino básico. É um portal Web destinado a crianças dos 4 aos 12 anos, que permite uma aproximação gradual às Tecnologias da Informação e Comunicação.

Uma escolinha é uma representação, na Internet, de uma escola real. Cada uma possui um espaço web privado e um espaço web público. No espaço privado é garantida a colaboração e

partilha entre professores, alunos e encarregados de educação dessa escola e o espaço público destina-se à publicação e exposição dos melhores trabalhos da escola.

A utilização da plataforma escolinhas.pt vai além de uma sala de aula ou escola, podendo ser acedida pelos utilizadores registados a partir de casa ou de outro lugar. Nela os alunos podem ler, escrever, pintar, desenhar, brincar, colaborar, partilhar e estar com colegas da escola, amigos, encarregados de educação e professores. Todas estas ações são realizadas de forma intuitiva e divertida, mas sem pôr em causa a segurança e a privacidade.

É objetivo da plataforma escolinhas.pt promover a correta utilização do software e boas práticas de ensino com as TIC.

1.2 A programação e os problemas do seu ensino a crianças

Esta secção vai abordar o tema da programação e os desafios que a mesma apresenta quando dirigida a crianças. Os problemas aqui identificados constituem a motivação para realização deste trabalho.

Atualmente a programação é vista como uma atividade técnica, restrita a um pequeno segmento da população. Este conceito da programação tem vindo a ser combatido, desde o aparecimento dos primeiros computadores pessoais, através do ensino a crianças de como programar [RMMH+09].

Aprender a programar pode ser muito difícil para iniciantes de qualquer idade, e essa dificuldade poderá tornar se ainda maior quando estamos a falar de crianças. São muitas as barreiras que colocam dificuldade ao ensino da programação a crianças [KP05]:

- **Estruturar a solução** — Aprender a solucionar os problemas de forma estruturada e bem definida;
- **Execução** — Aprender como os programas são executados;
- **Sintaxe** — Aprender a sintaxe e os comandos das linguagens;
- **Desmotivação** — Programar não é visto como uma atividade aliciante nem e divertida;

Na tentativa de ultrapassar essas barreiras, têm surgido desde então várias abordagens de ensino com o intuito de introduzir a programação a crianças (revisão no Capítulo 3). Contudo, a grande maioria deles acabaram por falhar ou serem abandonadas devido aos seguintes fatores: [RMMH+09]

- **Linguagem demasiado complexa** — As primeiras linguagens de programação detinham ainda uma sintaxe muita complexa, impedindo assim que muitas crianças as dominassem;
- **Inadequação dos Conteúdos** — A introdução da programação é feita ao mesmo tempo que se introduz conteúdos que não interessam as crianças. Ex: Escrever um programa para imprimir uma sequência de números primos;

- **Pouca profundidade** — Os contextos utilizados na introdução da programação a crianças, não fornecem uma orientação sobre o que fazer quando algo corre mal, nem uma maior exploração quando correm bem;

1.3 Objetivos

Este trabalho tem como objetivos o desenho e a prototipagem de um jogo para a introdução dos conceitos da programação a crianças. Pretende-se criar um mundo virtual onde as crianças, representadas por avatares, poderão interagir com os objetos presentes e com outras crianças, colaborando e resolvendo os desafios de programação que vão sendo propostos.

Os desafios de programação apresentados, são baseados numa nova linguagem de programação, adaptada para crianças no contexto do jogo, para crianças. Os desafios têm diferentes graus de dificuldade e incidem sobre diferentes conceitos da programação.

Para garantir interação e colaboração entre os jogadores, o jogo segue uma tipologia MMOG, onde os jogadores partilham o mesmo espaço, interagem uns com os outros e fazem convites, no sentido de resolverem em conjunto os desafios de programação.

1.4 Estrutura do Documento

O presente documento encontra-se dividido em sete capítulos.

Este inicial, a introdução, pretende apresentar ao leitor uma perspetiva geral do trabalho realizado, tendo em conta o problema, a motivação e os objetivos a serem alcançados.

Os capítulos 2 e 3 correspondem a uma revisão bibliográfica das principais áreas abrangidas neste trabalho, Jogos de computador e Linguagens/Ambientes de programação para crianças. No final de cada um destes capítulos é apresentado um resumo e uma breve conclusão de tudo que foi abordado e a influência que estes têm sobre este projeto.

No capítulo 4 é apresentado o desenho do jogo, Boobo World, o conceito, a mecânica, os objetivos, e outros aspectos importantes do jogo. Em seguida, o capítulo 5 descreve a implementação do protótipo do jogo e quais as funcionalidades que este contém.

Este protótipo foi testado com crianças, e os resultados obtidos estão no capítulo 6. Os resultados foram analisados e agrupados em gráficos e tabelas, facilitando assim a compreensão.

Por fim, o capítulo 7 faz uma conclusão do trabalho realizado tendo em conta a satisfação dos objetivos propostos e do trabalho futuro.

Capítulo 2

Jogos de Computador

Os jogos sempre foram populares, tanto no passado como no presente, e com a invenção dos computadores passaram a ser ainda mais persuasivos [SHO03].

Um jogo de computador pertence a um subconjunto do mundo dos jogos, os videojogos, Figura 2.1 [Wik11e].

Figura 2.1: Jogos de computador no mundo dos jogos

A definição de um jogo de computador está intimamente ligado à definição do que é um jogo. Um jogo possui pelo menos 3 componentes [SHO03]:

- **Jogadores** — Que estão dispostos a participar no jogo por gozo ou por diversão. Há respeito pelas regras do jogo e vontade de alcançar os objetivos.
- **Regras** — Que definem o jogo e consequentemente os objetivos deste;
- **Objetivos** — As metas que os jogadores têm de alcançar. Quando o jogo envolve mais do que um jogador os objetivos podem gerar competição e rivalidade entre estes.

Desta forma, um jogo de computador não é nada mais que um jogo realizado com a ajuda de um computador. Contudo, isto não significa que todos os jogos podem ser "convertidos" em jogos de Computador.

Outra abordagem formal para definir um jogo[1] passa pela análise da sua Mecânica-Dinâmica-Estética (do inglês *Mechanics-Dynamics-Aesthetics*, MDA) [Wik11d].

O MDA foi apresentado em 2001 por Robin Hunicke, Marc LeBlanc e Robert Zubek como uma framework para entender os jogos - esta abordagem estabelece uma ponte de ligação entre o *design* do jogo, o seu desenvolvimento, a sua crítica e os seus aspetos técnicos. Estes 3 componentes significam [HLZ04] [Gal09]:

- **Mecânica** descreve os dados e os algoritmos.
- **Dinâmica** descreve o comportamento do jogo. Relação entre as ações do jogador (*inputs*) e as respostas provenientes da sua mecânica (*outputs*).
- **Estética** descreve as respostas emocionais esperadas no jogador quando este interage com o sistema do jogo.

A perspetiva destas três componentes que o produtor do jogo tem é diferente da perspetiva de quem o vai consumir, o jogador. Podemos ver isto na Figura 2.2, do ponto de vista do produtor temos a mecânica a dar origem à dinâmica, que por sua vez, leva a um determinado tipo de experiências estéticas, e, do ponto de vista do jogador temos a estética a definir uma emoção, que resulta da observação de uma determinada dinâmica e, eventualmente, uma mecânica operacional [HLZ04].

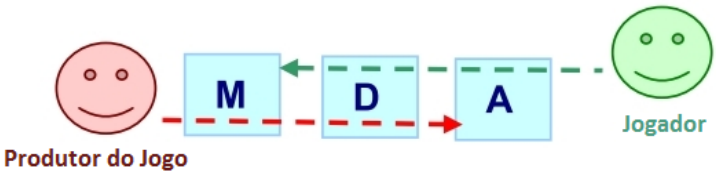

Figura 2.2: O *produtor* e jogador têm perspetivas diferentes

2.1 Géneros

Existem milhares de jogos[2] cada um com a sua própria configuração e arte. Embora não havendo um consenso na sua definição, os géneros dos jogos surgem, em *Game Design*, como um método para categorizar os jogos de acordo com a sua jogabilidade, interação e desafios apresentados ao jogador [Wik11f].

[1] jogo de computador/videojogo
[2] jogos de jogos de computador/videojogos

A seguir é apresentado na Tabela 2.1 um conjunto de géneros mais utilizados, as suas descrições e exemplos [Wik11f] [App06]:

Tabela 2.1: Géneros dos jogos de computador/videojogos.

Géneros	Caraterísticas	Exemplos
Ação	É dos géneros mais conhecidos. São jogos onde os reflexos, pontaria e timming dos jogadores para ultrapassar os obstáculos tornam-se a chave para alcançar a vitória. É composto por vários subgéneros tais como: jogos de luta, *first/third-person shooters*, jogos de plataformas, etc...	*Pong, Super Smash Bros. series, Call of Duty, etc...*
Estratégia	São todos os jogos que requerem cuidado especiais, habilidades e planeamento para que a vitória seja alcançada. Encontram se divididos em 2 subgéneros: estratégia em tempo real (do inglês Real Time Strategy, RTS) e estratégia baseada em turnos (do inglês Turned-Based Strategy, TBS), possuindo uma estética, visão geral das ações em curso e representação dos cenários reais similares.	*Globulation 2, Bos Wars, Age of Empires series, etc...*
Simulação	Corresponde a uma categoria muito diversificada no mundo dos jogos e está relacionado com a simulação de aspetos da vida real ou fictícia. São jogos que simulam desportos, condução de veículos (aviões, carros,... etc), dinâmicas de vilas, cidades ou pequenas comunidades.	*Trucks and Trailers, Farmville, etc...*
Role-playing	O género role-playing games(RPG) esta intimamente ligado à literatura/mundo da fantasia. Incluí subgéneros como *action RPG* que engloba elementos dos jogos de ações, *Massively multiplayer online role-playing games (MMORPGs)* cuja principal caraterística é a interação entre muitos jogadores num mundo virtual (mundo da fantasia).	*Grand Theft Auto series, World of Warcraft, etc...*

Além destes géneros mais conhecidos, existem no mundo dos jogos outros com caraterísticas próprias, ou que combinam elementos de vários géneros, como por exemplo o género aventura, ou ações e aventura.

2.2 Serious Games

A indústria de jogos está essencialmente virada para o entretenimento, e a importância que esta área tem atualmente pode ser encontrada nas seguintes palavras de Kojima[3]:

[3]Hideo Kojima - Game producer

"Os videojogos são, nos dias de hoje, a mais rentável indústria de entretenimento, superando os filmes em termos de receitas. Mas, quando comecei, a história era completamente diferente. Nessa altura, a indústria dos videojogos era um lugar de onde as pessoas com sonhos desfeitos se reuniam pois não conseguiam o emprego que realmente ambicionavam. Hoje, a indústria amadureceu, tornando-se um lugar maravilhoso onde algumas das pessoas mais talentosas na sua área podem tirar vantagem de tecnologias de ponta e investimentos astronómicos para emocionar e admirar o planeta inteiro [myG11]."

No entanto, nem todos os jogos são desenvolvidos tendo o entretenimento como objetivo principal. Estes objetivos são variáveis, tendo em conta a natureza da informação que pretendem transmitir, persuadir ou simular [Wik11f]. Como exemplo temos os *Serious Games*.

O Conceito de *Serious Game*, em português "Jogo sério", esta intimamente o ligado ao conceito de um *Jogo* diferenciando-se deste apenas no objetivo a alcançar.

Um jogo tem como objetivo principal garantir entretenimento de quem o joga, tem uma história, uma arte e uma interação.

Um *Serious Game* é desenhado com o objetivo de melhorar algum aspeto de um processo de aprendizagem [Der07].

É apresentado na Figura 2.3 uma definição de *Serious Game*, onde podemos verificar que para além das características de um jogo, *story*, *art* e o próprio *software* um *Serious Game* possui uma componente pedagógica que se pretende transmitir ao jogador.

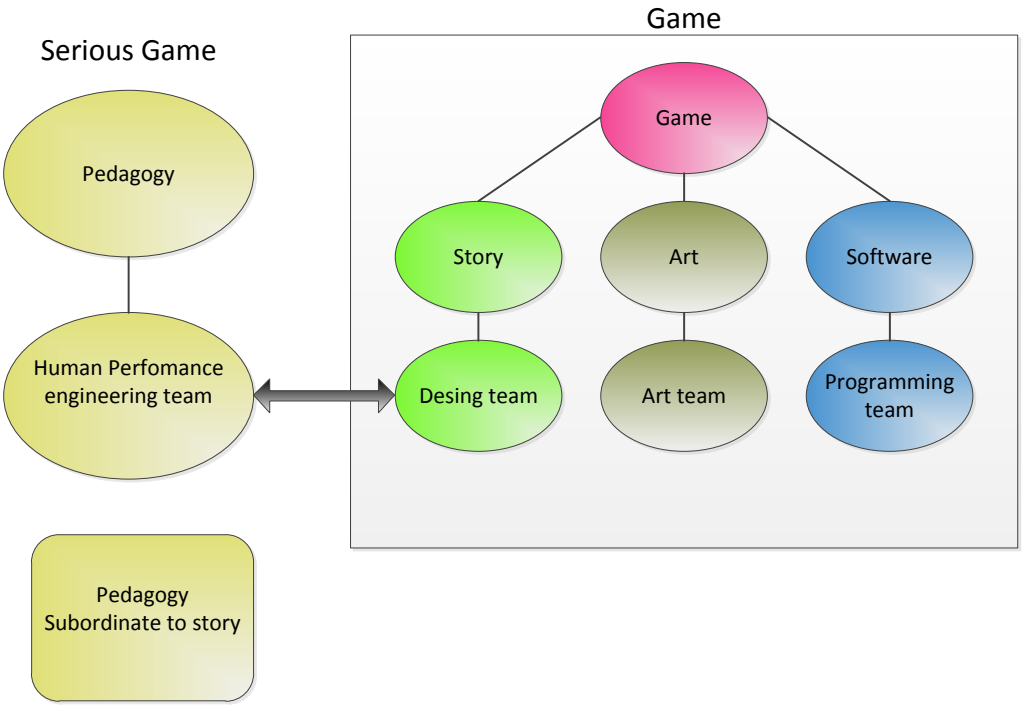

Figura 2.3: Diferença entre um *Serious Game* e um *Game* [QN08]

A utilização dos *Serious Games* estende-se às mais diversas áreas e atuam de várias maneiras. Na Figura 2.4 temos uma taxonomia dos *Serious Games* resultado da associação entre essas áreas e os objetivos a serem alcançados [SS08].

	Games for Health	Advergames	Games for Training	Games for Education	Games for Science and Research	Production	Games as Work
Government & NGO	Public Health Education & Mass Casualty Response	Political Games	Employee Training	Inform Public	Data Collection / Planning	Strategic & Policy Planning	Public Diplomacy, Opinion Research
Defense	Rehabilitation & Wellness	Recruitment & Propaganda	Soldier/Support Training	School House Education	Wargames / planning	War planning & weapons research	Command & Control
Healthcare	Cybertherapy / Exergaming	Public Health Policy & Social Awareness Campaigns	Training Games for Health Professionals	Games for Patient Education and Disease Management	Visualization & Epidemiology	Biotech manufacturing & design	Public Health Response Planning & Logistics
Marketing & Communications	Advertising Treatment	Advertising, marketing with games, product placement	Product Use	Product Information	Opinion Research	Machinima	Opinion Research
Education	Inform about diseases/risks	Social Issue Games	Train teachers / Train workforce skills	Learning	Computer Science & Recruitment	P2P Learning Constructivism Documentary?	Teaching Distance Learning
Corporate	Employee Health Information & Wellness	Customer Education & Awareness	Employee Training	Continuing Education & Certification	Advertising / visualization	Strategic Planning	Command & Control
Industry	Occupational Safety	Sales & Recruitment	Employee Training	Workforce Education	Process Optimization Simulation	Nano/Bio-tech Design	Command & Control

Figura 2.4: Taxonomia dos *Serious Games* [SS08].

De acordo com esta taxonomia, este projeto é sobre a criação de um jogo do tipo *Learning game* ou *Digital Game-Based Learning*.

Uma condição essencial para uma aprendizagem de sucesso é a motivação. Um aprendiz motivado não pode ser parado, estará sempre focado e com vontade de aprender cada vez mais [Pre03].

Contudo, a motivação é o maior problema presente em todas formas de aprendizagem formais, tanto numa sala de aula, à distância ou através do *e-learning*, há sempre uma necessidade de manter os estudantes motivados o suficiente para que estes possam terminar com sucesso qualquer tarefa que tenha o processo de aprendizagem incluído, como por exemplo terminar uma aula, um curso, um semestre ou um ano letivo. Isto acontece porque aprender requer esforço e como tal, sem motivação raramente as pessoas estão dispostas a fazê-lo [Pre02].

Por outro lado, temos os jogos de computador/videojogos que prendem o jogador por horas e horas afins, sem muitas vezes se aperceberem disso [Pre02]. Esta motivação só é alcançada se os jogos de computador apresentarem as seguintes caraterísticas [MSS04] [Tee08]:

- **Desafio contínuo**— O jogador tem que sentir constantemente desafiado para poder continuar a jogar. Cada desafio deve ter um objetivo e posteriormente levar o jogador a avançar no jogo superando os desafios seguintes.

- **Guião interessante**— Depende do tipo do jogo, em jogos que temos apenas um jogador num mundo virtual, uma boa história motiva o jogador a ter sucesso no jogo. Em jogos com mais de um jogador, e onde competem uns com os outros, a história não é essencial, já que a motivação provém da competição.
- **Flexibilidade**— Num jogo cada objetivo deverá ter várias formas de ser alcançado. Fornecendo assim ao jogador a possibilidade de criar várias estratégias para alcançá-los e ao mesmo tempo manter o jogo desafiante.
- **Recompensas uteis imediatas**— Para além da vitória final, um jogo deverá ter outro tipo de recompensas para os jogadores que vão completando os desafios com sucesso. Estes prémios podem ser novas capacidades, novas armas, ou mesmo novos desafios.
- **Combinar diversão e realismo**— Um jogo deverá combinar simulações reais com divertimento. Jogos onde a simulação da realidade é mais importante que os objetivos e os desafios, ou onde o jogador tem que perder muito tempo a tomar decisões, podem acabar por se tornar chatos.

Desta forma *Digital Game-Based Learning* é precisamente sobre divertimento e fazer participar(motivar) a pessoa, trazendo ao mesmo tempo uma aprendizagem séria [Pre01].

Os jogos para educação, abrangem tantos adultos, jovens e crianças e destinam-se tanto a aprendizagem formal como informal[SS08]. Como exemplos deste tipos de *Serious Games* temos o ***Flight Simulator*** e o ***Genomics Digital Lab***.

Flight Simulator É uma recriação artificial do voo das aeronaves e do ambiente que o rodeia. Proporciona ao jogador uma simulação rica e realista, Figura 2.5, com vários aviões utilizados em missões interativas ou situações de voo simples. O ambiente que rodeia os voos inclui equações que controlam o comportamento do avião e como este reage aos controlos [Wik11a][Cor06].

Figura 2.5: Interface Flight Simulator.

Genomics Digital Lab Este jogo, Figura 2.6, pretende ensinar a estudantes do secundário conceitos de biologia como fotossíntese, respiração, transcrição e translação. É composto por vários níveis, cujo o grau de dificuldade vai aumentado à medida que o jogador avança no jogo. Por exemplo no início o jogador é confrontado com um desafio de identificar quais condições necessárias para manter uma planta viva, enquanto que os níveis mais avançados são baseados em tópicos do mundo real [Wik11b] [Int10].

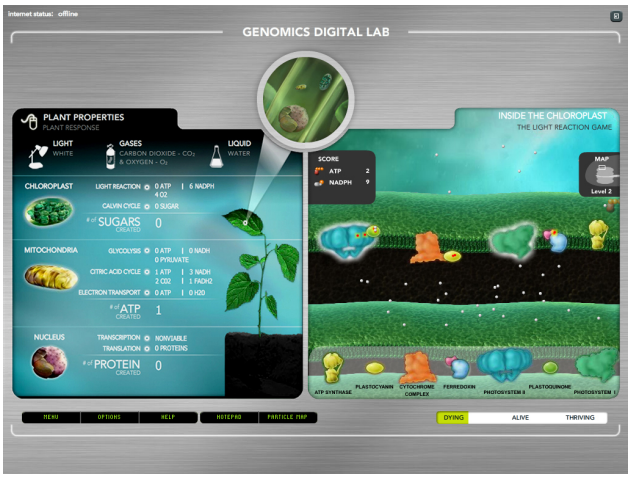

Figura 2.6: Interface Genomics Digital Lab.

Existem também jogos como o *ClickN READ Phonics* e *ClickN SPELL* que ensinam as crianças a ler e a soletrar respetivamente.

2.3 Componente Multiplayer e Colaboração

Um jogo [4] pode ser jogado por vários jogadores, *Multiplayer Game*. Os jogadores podem optar por jogar sozinhos, contra outros oponentes, equipas contra equipas, ou uma equipa contra o computador. Como exemplos deste tipo de jogos temos os *Massively Multiplayer Online Games* (MMOG) [Wik11c].

Massively Multiplayer Online Games Massively Multiplayer Online Games(MMOG) são um género de jogos de computador caraterizados por um mundo virtual (2D ou 3D) online, onde o jogador tem a possibilidade de criar personagens digitais, os avatares, e posteriormente, interagir com esse mundo e com outros jogadores. Estes mundos virtuais incluem a representação de florestas, cidades, mares, castelos, etc... Os MMOGs têm tido uma grande evolução nos últimos anos na indústria do entretenimento, ganhando popularidade entre as pessoas de qualquer faixa etária, etnias, ou classes económicas [Ste04]. Estes jogos são construídos com o objetivo de ligar as pessoas num espaço virtual, permitindo assim a criação de equipas, partilha de informações e ao mesmo tempo assegurar a colaboração na resolução dos desafios [BD05] [LW08].

[4] jogo de computador/videojogos

Quando se fala de colaboração também é frequente falar de cooperação [PI11], porém existe uma pequena diferença entre estes dois conceitos [Bai89] [Pan97]:

- **Colaboração** — Corresponde a participação num mesmo processo, como por exemplo preparar um memorando, um formulário ou desenvolver um software. Embora possa haver uma saída comum, cada individuo é avaliado de forma independente pois a interação e participação nem sempre é igual.

- **Cooperação**— Os objetivos do individuo são substituídos pelos objetivos do grupo. Há uma saída comum a todos e a avaliação é feita para toda equipa.

A colaboração em jogos de computador(género MMOG) segue uma estratégia de exploração. Os jogadores descobrem de forma partilhada um local, uma atividade, um ambiente ou um tópico. Esta descoberta é feita nos mundos virtuais que compõe o jogo [NK08] [IRW+10].

Como exemplos de MMOGs destinados a crianças temos o *Moshi Monsters*, *Club Penguin* e o *Driving kids*.

Moshi Monsters Moshi Monsters é um jogo (Figura 2.7) online gratuito para as crianças, onde eles podem adotar um monstro e cuidar dele.

Figura 2.7: Interface do Moshi Monsters.

As crianças cuidam dos seus monstros resolvendo puzzles, que dão recompensas ao seu monstro virtual. O monstro tem associado um conjunto de itens virtuais como móveis, brinquedos, comida e guloseimas. Com o passar do tempo, o monstro passa para um outro nível, onde poderá visitar novos locais no mundo virtual(uma cidade de monstros) e ganhar novas recompensas e novos jogos. Cada criança juntamente com o seu monstro pode fazer amizades com outros utilizadores e deixar mensagens nas suas páginas.

O Moshi Monsters é também um jogo educativo, pois todos os dias as crianças são confrontados com novos desafios de vocabulário, aritmética, lógica e habilidades especiais. A dificuldade dos desafios vai aumentando a medida que eles vão resolvendo os de forma correta [Ltd08]. Para além destes desafios, as crianças também têm que aprender a alimentar os seus monstros de forma eficaz, através do desenvolvimento de estratégias, gestão de recursos, colaboração com os amigos e outras habilidades.

Outra caraterística do Moshi Monsters é a comunicação que existe entre o monstro e o seu dono, através de balões de fala que incentivam a leitura, e também o humor do monstro("eufórico", melancólico, etc...) são representados gráficamente [Ltd08].

Club Penguin O Club Penguin é um mundo virtual, uma ilha coberta de neve,(Figura 2.8) para crianças. Cada criança tem um avatar pinguim colorido e com o qual junta-se à comunidade e envolve-se numa variedade de atividades e eventos. Pode ainda conversar com os amigos através do chat.

Os jogos existentes na ilha servem para ganhar moedas virtuais que podem ser utilizados para criar uma casa (igloo) e outros acessórios para o pinguim [Inc11].

Figura 2.8: Interface do Club Penguin.

Driving Kids Este jogo é um MMOG destinado a crianças do pré-escolar e 1º ciclo e tem como objetivo, ser educacional. Incluí um conjunto de minijogos e atividades para ajudar as crianças no processo de aprendizagem digital.

Driving Kids passa-se num mundo virtual, Figura 2.9, onde cada criança com o seu avatar, vai explorando o mundo, deslocando para novos lugares, interagindo com os objetos do mundo e socializando com os restantes jogadores [Kid12].

Figura 2.9: Mundo virtual do Driving Kids

Os minijogos presentes no Driving Kids são baseados em séries de jogos para educação, existentes, que tem como objetivo ajudar as crianças a melhorar as suas capacidades de memorização, resolução de problemas, desenvolver pensamentos criativos e de lógica, melhorar os seus conhecimentos de matemática entre outros [Kid12].

Um exemplo destes jogos, é um puzzle de memória, Figura 2.10, que no Driving Kids consiste em encontrar os camiões com a mesma combinação de cores.

Figura 2.10: Um puzzle de memória no Driving Kids

2.4 Desenho de um Jogo

As pessoas quando pensam na indústria de videojogos imaginam sempre um mundo de diversão, muita festa e pouco trabalho. Porém, pensar, desenhar, criar e publicar um jogo são tarefas que exigem muito estudo, esforço, dedicação e motivação. Existe um conjunto de "estados" pelo qual o jogo tem de passar até chegar as mãos do jogador final. Esses estados são [Ord11]:

2.4.1 Ínicio

O ponto de partida, *Brainstorming*, para criar qualquer jogo é pensar sobre o mesmo. Neste estado, todos os intervenientes na criação do jogo dão as suas ideias, começam por dar ideias para o jogo, e por mais absurdas que possa parecer elas são aceites por todos sem qualquer crítica.

Após serem recolhidas todas as ideias, passa-se para uma fase de triagem onde são eliminadas as ideias irrelevantes. Todas as ideias que permanecem depois desta fase, são organizadas em grupos.

Por fim são atribuídas prioridade aos grupos e seleciona-se o melhor.

2.4.2 O primeiro Documento

Após a fase de *Brainstorming* é necessário focar na ideia geral do jogo. Para isso procede-se a escrita de um documento de alto nível, "Game Design Document". Este documento deverá conter uma descrição dos componentes do jogo presentes na Figura 2.11.

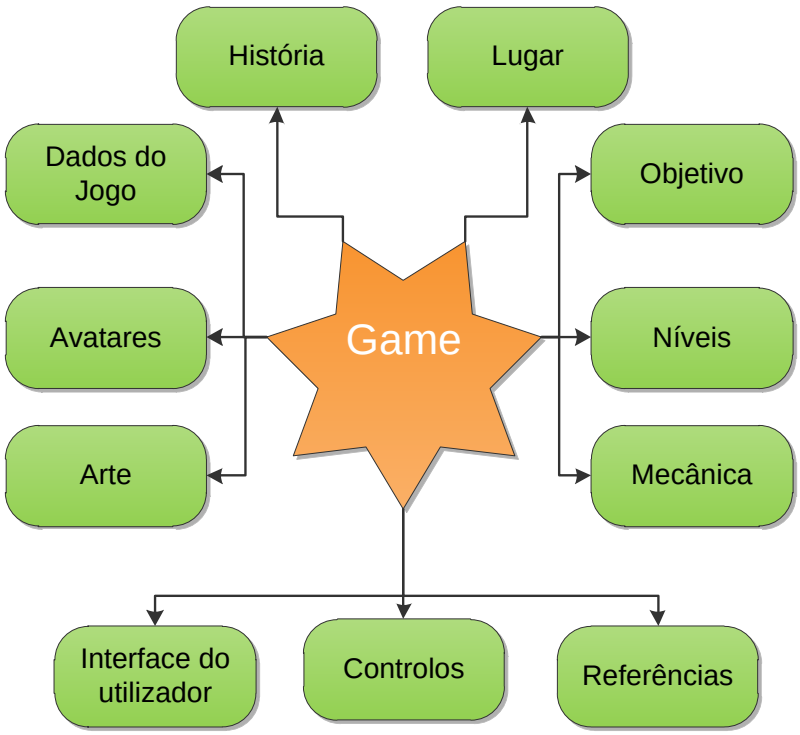

Figura 2.11: Componentes de um documento de desenho do jogo.

Dados do jogo: Neste capítulo do documento deve constar informação mais alto nível do jogo como o título, a plataforma de destino, a tipologia, se é um jogo *single Player* ou *Multiplayer*, o público alvo entre outros.

História: Nesta secção do documento faz-se uma contextualização do jogo e uma introdução do lugar onde onde o jogo vai desenrolar-se. Esta contextualização deve ser "contagiante" de modo a cativar o jogador.

Lugar: É necessário fazer uma descrição do espaço onde o jogo vai desenrolar-se, espaço físico, mundo da fantasia, ... etc.

Objetivo: Em todos os jogos, um jogador deverá saber à partida o que tem de fazer. Esta informação define os objetivos do jogo.

Mecânica: Depois de definir quais os objetivos do jogo, é preciso definir um conjunto de regras e condições que devem ser cumpridas para que o jogador atinga estes mesmos desafios. Estas devem ser compreendidas por todos os intervenientes do jogo, desde o criador até ao jogador.

Níveis: Os níveis correspondem a espaços específicos no jogo, são lugares ondem acontecem determinadas ações, possuem um ponto de interesse, o jogador tem de cumprir um determinado objetivo, poderá existir um caminho que o avatar do jogador terá de seguir e pode ser *single Player* ou *Multiplayer*.

Avatares: Os avatares são uma representação virtual do jogador no mundo do jogo. Alguns dos atributos que estes podem ter são: nome, idade, sexo, descrição física, forças, fraquezas, motivação. Os avatares costumam também ter uma pequena história associada. Esta história tem como objetivo contextualiza-lo no jogo.

Arte e Som: Também é importante deixar algumas pistas para os artistas do jogo. Estas pistas servem como um ponto de partida para o trabalho que irão realizar.

Interface do Utilizador: Consiste em listar e descrever quais as informações que os jogadores têm disponível no jogo (no ecrã) e com as quais podem interagir.

Controlos: Secção destinada a descrever como os jogadores vão interagir com o jogo.

Referências: O jogo é um *remake*, ou continuação de outro jogo conhecido? Caso seja, incluir essa referência no documento.

2.4.3 Ferramentas

Estando o primeiro documento concluído, é necessário proceder a escolha das ferramentas que ajudam na criação do jogo. Isso inclui a escolha da tecnologia, motores de jogo e mesmo ferramentas de aumento da produtividade da equipa. Essa escolha deverá ser sábia e adequada ao contexto, por exemplo para fazer um jogo em 2D não é necessário uma ferramenta que faça jogos em 3D.

2.4.4 Riscos

Como qualquer outro projeto de criação de software, existe um conjunto de riscos que devem ser analisados quando se esta a começar a criação de um jogo. Alguns destes riscos são: a dificuldade da tecnologia, o código será muito complexo, é preciso muitos elementos gráficos, em caso de jogos sérios será preciso um treino especial, simulações requerem uma experiência na vida real,... etc.

2.4.5 Lançamento do Jogo

Por fim é preciso analisar as razões que levarão o jogo a ter sucesso quando for lançado. Estas razões podem ser:

- Inovação na forma de jogar.
- O desenvolvimento do jogo foi barato.
- A audiência do jogo é grande.
- A equipa que desenvolveu o jogo é famosa, e isso vai atrair atenção de todos.
- O jogo é uma sequela de um jogo conhecido e bem sucedido.

As fases descritas anteriormente não são rígidas e variam de jogo para jogo. Cabe aos criadores do jogo decidir quando e como passar por cada uma delas.

2.5 Resumo

Atualmente as crianças têm grande facilidade em interagir com as novas tecnologias e sobretudo, não há nada que prende mais a atenção de um uma criança do que um jogo. Além do entretenimento que um jogo fornece, há uma motivação (recompensas, prestígio, pontuação, ... etc) e disponibilidade para continuar a jogar até alcançar todos os desafios que o jogo possa apresentar.

Contudo um jogo não serve apenas entretenimento, este pode ser utilizado como meio para o ensino e aprendizagem de uma determinada temática. Neste sentido a criação de um *Serious Game* para o ensino da programação surge como uma opção viável para satisfação os objetivos pretendidos nesta dissertação. Porém, antes da criar um jogo é necessário desenhá-lo, e neste capítulo temos um primeiro contato com o mundo dos jogos, quais os seus objetivos, e como desenhar um jogo.

Capítulo 3

Linguagens e Ambientes de Programação para Crianças

Desde o aparecimento dos primeiros computadores pessoais que têm surgido vários ambientes e linguagens de programação para crianças. Todas são uma nova tentativa de tornar a programação mais acessível às crianças. Segundo Papert, esta acessibilidade só é conseguida se as linguagens de programação respeitarem três princípios [RMMH+09]:

1. **"Low floor"** — Fácil de começar a utilizar.
2. **"High ceiling"** — Fornece oportunidades para criar de forma incremental projetos mais complexos com o passar do tempo.
3. **"Wide walls"** — Capacidade de suportar diferentes tipos de projetos, de forma que pessoas com interesses e estilos de aprendizagem diferentes se sintam motivadas.

As diferentes linguagens e ambientes de programação que têm surgido nos últimos anos podem ser classificadas como pertencentes a um dos seguintes grupos, tendo em conta o seu objetivo principal [KP05]:

- **Linguagens e Sistemas de Ensino da Programação** — Estes sistemas são desenhados com o objetivo de ajudar as pessoas a aprender a programar. No entanto os sistemas presentes nesta categoria apresentam dois objetivos, conflituosos, quando são criados: um sistema fácil de usar para os utilizadores iniciantes à programação, ou um sistema capaz de fornecer uma base sustentável para uma linguagem de programação mais avançada. A solução passa por estabelecer um equilíbrio entre os dois objetivos.
- **Programação como Suporte/Capacitação para outras áreas** — Ao contrário dos sistemas na categoria anterior, aqui não há preocupação com a transição dos conhecimentos adquiridos para uma linguagem de programação mais avançada. Os projetistas deste tipo

de sistemas acreditam que o aspecto mais importante da programação é permitir as pessoas criarem coisas que se adaptam as suas necessidades. Conseguir que as pessoas produzam mais e melhor é a principal preocupação destes sistemas.

Em ambas categorias há um objetivo claro, tornar a programação mais acessível às crianças ou qualquer outra pessoa que esteja a ter o primeiro contato com a programação. No entanto dentro de cada categoria há diferentes objetivos, por exemplo nos sistemas de ensino a grande maioria foca-se no ensino dos mecanismos de programação, que é a área de maior dificuldade para os iniciantes, ensinar a exprimir intenções para o computador e entender as ações deste. Outros sistemas tentam dar razões concretas para programar ou tentam suportar o trabalho em equipa e a aprendizagem colaborativa entre os programadores mais novos/inexperiente [KP05].

3.1 Linguagens e Sistemas de Ensino da Programação

Os sistemas de ensino, como o próprio nome indica, pretendem ensinar os mecanismos da programação. Na Figura 3.1 podemos ver os objetivos específicos que estes sistemas pretendem atingir. Ensinar a criar, ensinar a estruturar e compreender a execução [KP05].

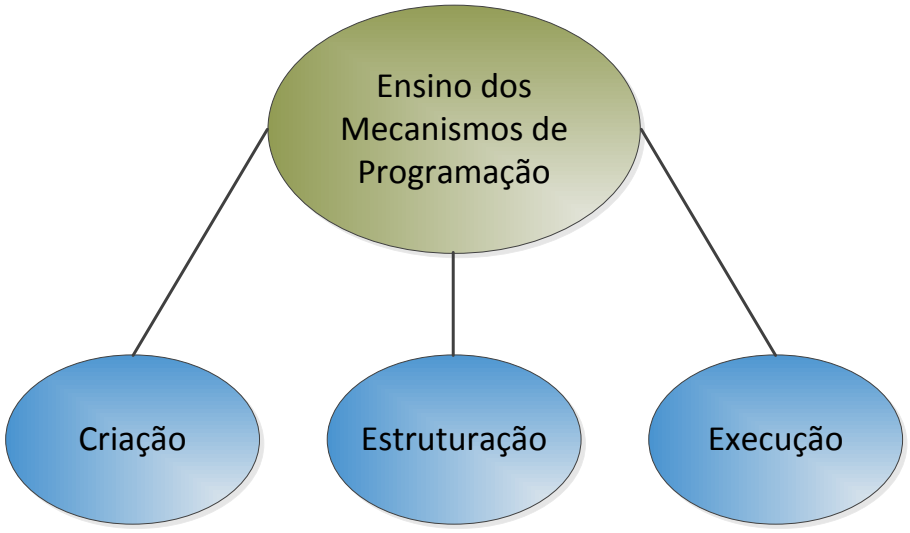

Figura 3.1: Linguagens e Sistemas de Ensino dos mecanismos da Programação

3.1.1 Ensinar a Criar

Para criar um programa na grande maioria das linguagens é necessário a introdução de texto num editor. Para um programador iniciante, criança ou não, há sempre o problema de conseguir traduzir as suas intenções, de forma correta, para o computador. Há duas alternativas, 1) Simplificar a introdução do código, simplificando a linguagem, adequar a linguagem a um domínio restrito ou prevenir os erros de sintaxe, 2) Encontrar alternativas a digitação do código. [KP05].

As formas alternativas ao tradicional método de programação, são as mais relevantes pois apresentam um maior atrativo para as crianças. Na Tabela 3.1 encontra-se descrito as diferentes abordagens utilizados por estes sistemas, as suas caraterísticas e alguns exemplos.

Tabela 3.1: Alternativas a Digitação do Código

Abordagens	**Caraterísticas**	**Exemplos**
Gráficos ou objetos Físicos	Substituição do código por objetos gráficos ou físicos, cada um representando um comando, um estrutura de controlo ou variáveis.	*LogoBlocks, Alice, Greenfoot, Scratch, Kodu.*
Interfaces de Ações	Utilização do resultado das ações realizadas na interface para criar um programa.	*LegoSheets*
Múltiplos Mecanismos	Múltiplos métodos para criação dos programas, o tradicional código, manipulação direta ou preenchimento de formulários.	*Leogo*

Alice

Alice é um ambiente de programação 3D inovador que facilita a criação de uma animação para contar uma história, jogar um jogo interactivo, ou criar um vídeo e depois partilhar na web [Ali10].Alice é uma ferramenta de ensino, destinada aos mais novos, cujo objetivo é fazer uma primeira exposição à programação orientada a objetos. Permite aos estudantes aprender os conceitos fundamentais da programação no contexto da criação de vídeos animados ou jogos simples [CD00] [Ali10].

Em Alice, os objetos 3D (ex. pessoas, animais e veículos) povoam um mundo virtual onde os estudantes podem criar programas para os animar. Estes programas são scripts simples, que durante a sua execução permitem aos objetos receber os *inputs* do utilizador via rato ou teclado e depois cada ação é animada suavemente durante o período de tempo especificado [CD00].

Alice possui uma interface, ver Figura 3.2, interativa constituído por peças que os estudantes podem manipular para construir um programa. Estas peças contêm instruções, padrões, de uma linguagem de programação orientada a objetos (java, C++ ou C#).

Como linguagem de programação, as principais características de Alice são [CD00]:

- **Ações (Transformadores de estados)** — Alice fornece um conjunto de comandos de ações que podem ser subdivididos em duas categorias: os comandos que dizem ao objeto para realizar um movimento e aqueles que mudam a aparência do objeto.

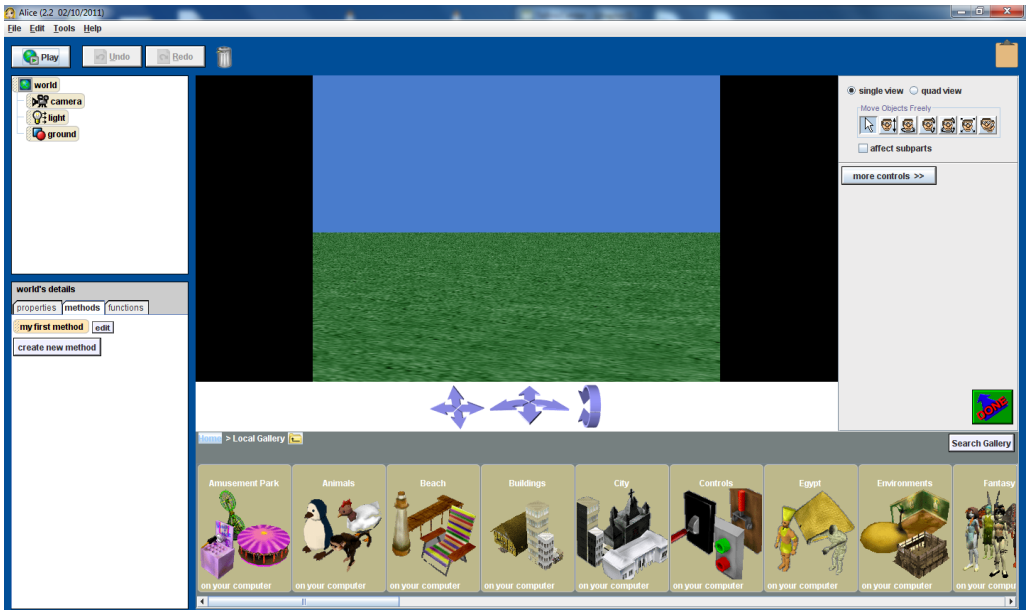

Figura 3.2: Interface do Alice.

- **Nomeação de instruções (Procedimentos)** — Alice permite nomear uma sequência de instruções. Este conceito é similar ao conceito de procedimentos em muitas outras linguagens de programação.
- **Funções** — Em Alice, as funções desempenham 3 papéis distintos: implementação da recursividade, implementação das interações via eventos e são úteis na computação.
- **Decisões** — As decisões em Alice correspondem aos comandos das ações que os objetos devem efetuar.
- **Recursividade** — Alice fornece suporte a repetição através de uma instrução de *Loop*.
- **Eventos/Interacções** — Alice suporta handlers para eventos e criação de interfaces gráficas de utilizador(GUI) com painéis de controlo, *list boxes*, *check boxes* e *sliders*.

Atualmente a versão do Alice - Alice 2, surge após o melhoramento de versões anteriores deste sistema (Alice98, Alice99) e serve como base para outras linguagens e ambientes de programação para crianças designado por ***MAMA*** [Sci10] [KP05].

Greenfoot

Greenfoot é um sistema composto por duas partes: A primeira parte, um ambiente integrado de desenvolvimento pedagógico concebido para ser fácil de usar por iniciantes. Como tal, este integra as ferramentas comuns (como um editor, compilador, máquina virtual), com ferramentas educacionais (como a invocação de objetos interativos, a inspeção, a visualização da classes). E a segunda parte, uma framework de simulação de um micro-world [KÖ8].

O ambiente de Greenfoot fornece uma plataforma que permite uma grande interação visual e aliciante para introdução da programação orientada a objetos em java. Os estudantes que utilizam este sistema concentram a sua atenção na programação do comportamento do objeto. O código em Greenfoot é apresentado de forma gráfica, desta forma os estudantes focam-se nos conceitos importantes em vez de preocuparem-se com a sintaxe da linguagem. Greenfoot tem uma interface visual, Figura 3.3, que fornece feedback imediado aos utilizadores.

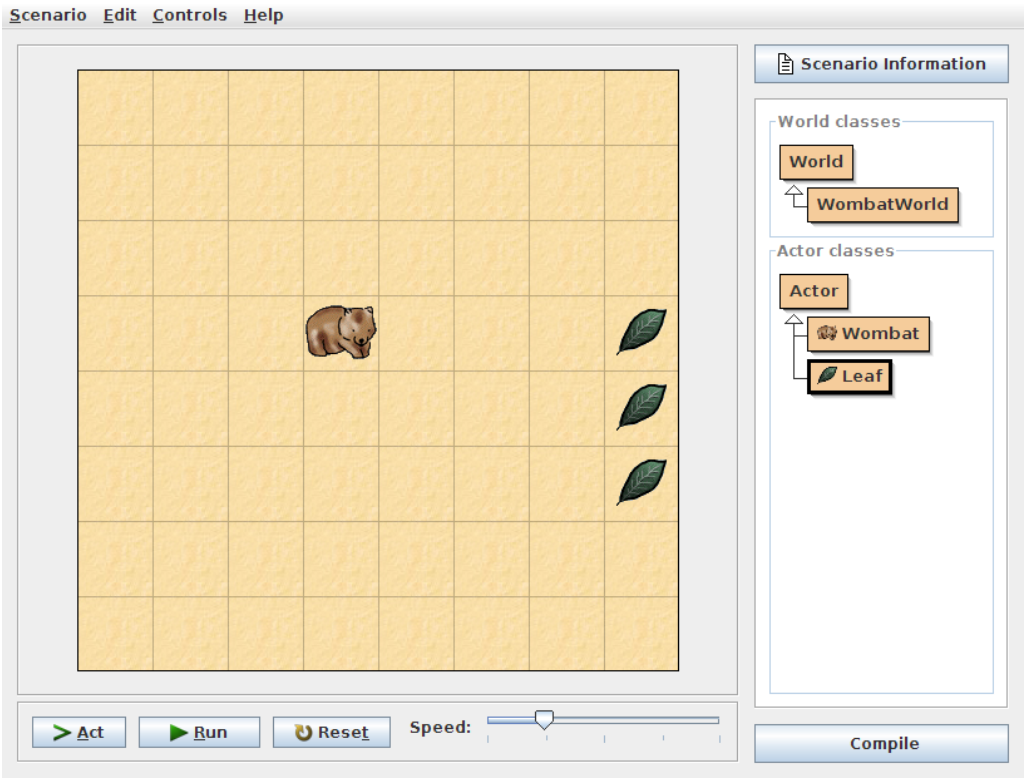

Figura 3.3: Interface do Greenfoot.

Este sistema pode ser utilizado em vários níveis da educação: ensino secundário, cursos de introdução a programação (do inglês CS1 courses) ou mesmo cursos avançados [HK04] [KÖ8].

Scratch

Scratch é um ambiente de programação visual, Figura 3.4, que permite aos utilizadores aprender programação enquanto trabalham em projetos pessoais do seu interesse, como histórias animadas ou jogos. Esta ferramenta foi desenvolvida tendo como alvo principal crianças entre os 8 e 16 anos de idade, mas Scratch tem como objetivo principal introduzir a programação à todas as pessoas que nunca imaginaram a si mesmos como programadores [MRR+10] [RMMH+09].

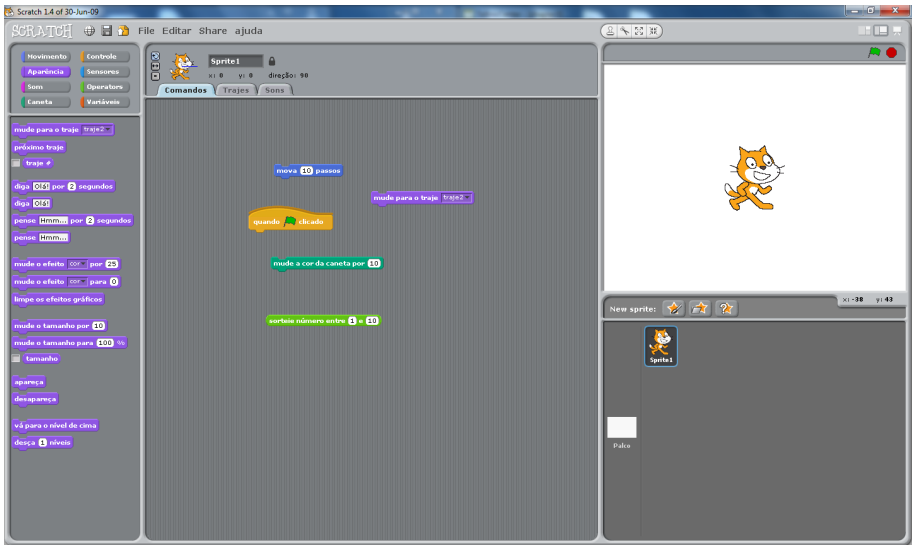

Figura 3.4: Interface do Scratch.

Tendo em conta este objetivo, o Scratch pode ser analisado de duas formas:

1. **Ambiente de Programação**— O ambiente de programação do Scratch pretende encorajar a auto-aprendizagem, convidando o utilizador a experimentar os *scripts* e recebendo feedback ao mesmo tempo. Toda esta interação encontra-se facilitada, pois o Scratch possui uma interface do utilizador baseada na filosofia - Single-Window User Interface [MRR+10]. Com isto todos os componentes chaves do Scratch estão sempre visíveis.

2. **Linguagem de Programação**— A programação em Scratch é feita juntando blocos coloridos, que contêm instruções, para controlar objetos gráficos em 2D, chamados sprites. Movendo os blocos sobre um fundo,*stage*, o utilizador cria os seus programas de forma interativa, e para começar a execução, basta clicar sobre a pilha de blocos criada [RMMH+09].

   O conjunto de blocos empilhados, constituem os *scripts* que representam declarações, expressões e estruturas de controlo. Cada bloco tem uma forma própria, ajudando assim o utilizador a encaixá-los, através de *drag-and-drop*, uns com os outros. Isto evita a criação de *scripts* que não façam sentido [MRR+10].

Kodu

Kodu é uma nova linguagem de programação visual feita especificamente para a criação de jogos e destinada, especialmente, às crianças. O ambiente de programação funciona tanto na Xbox e PC Windows. Este ambiente tem uma interface de programação que usa uma linguagem simples e totalmente baseado em ícones, Figura 3.5.

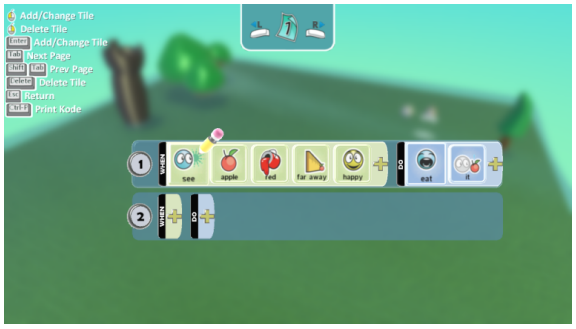

Figura 3.5: Kodu- Ambiente de programação e linguagem.

Kodu permite a programação de cada personagem e objeto individualmente para interagir com o mundo, podendo funcionar como agentes inteligentes. Os programas são expressas em termos físicos, usando conceitos como visão, audição, e tempo para controlar o comportamento do personagem. Além disso, Kodu tem outras características como construtor de pontes e caminhos, editor interativo, e 20 personagens diferentes com diferentes habilidades [Res11].

Kodu possui uma comunidade onde os utilizadores podem compartilhar seus projetos. Estes podem ser depois acedidos pelo Laboratório de Jogo Kodu, Figura 3.6, ou através de um site, chamado Planeta Kodu. No planeta Kodu os alunos também podem participar de competições, designadas por *Kodu Cups* [Res11] [Sto10].

Figura 3.6: Kodu Game Lab - Acesso aos projetos partilhados.

3.1.2 Ensinar a Estruturar

A principal preocupação dos sistemas nesta categoria é estruturação e organização do código. Não existe grande preocupação com sintaxe.

Neste grupo temos linguagens e ambientes de programação que tentaram criar novos paradigmas para programação(***Pascal, Smaltalk, Playground, Kara***, ...etc) e outros que tentaram tornar estes novos modelos mais acessíveis(***Liveworld, BlueJ, Karel++, Karel J Robot, J karel, ... etc.***) [KP05].

Kara

Kara ilustra os conceitos básicos de programação através da utilização de máquinas de estados finitos num ambiente gráfico, tornando assim num sistema atraente para iniciantes. Kara é uma joaninha, ver Figura 3.7, que vive num mundo simples, e cabe ao estudante programá-la para executar um conjunto de tarefas, como por exemplo recolher folhas [ST05]. Kara é baseado em Karel[KP05].

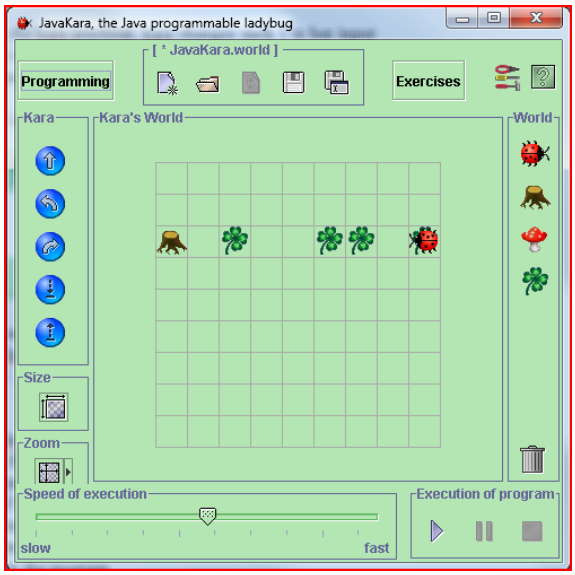

Figura 3.7: Interface do Kara.

Liveworld

É um ambiente programação desenhado para a construção de sistemas animados. Tem como objetivo fornecer um mundo onde todos os objetos podem ser facilmente manipulados. Tem um espaço virtual, onde os objetos estão hierarquicamente estruturados, designados por *boxes* (Figura 3.8). A linguagem de programação utilizada é o *Lisp* [Tra96].

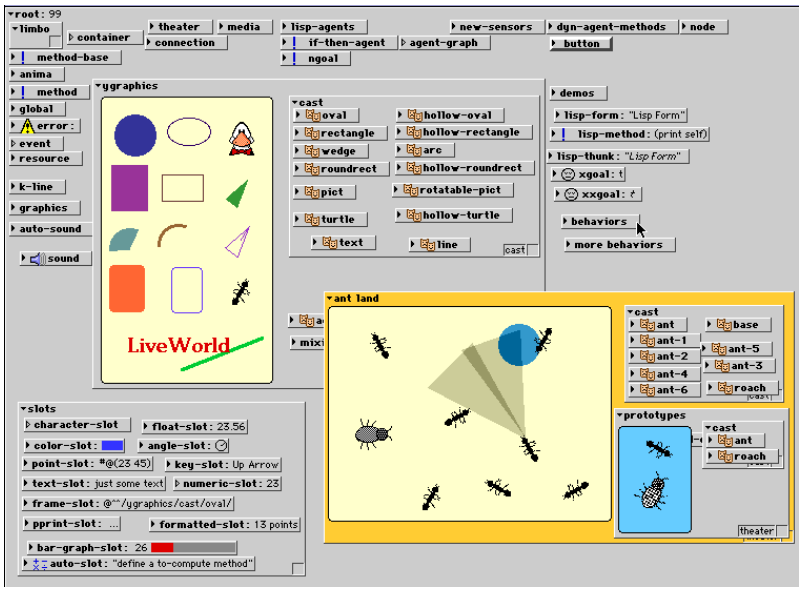

Figura 3.8: Interface do Liveworld.

3.1.3 Compreender a Execução

Os sistemas nesta categoria pretendem dotar os utilizadores iniciantes na programação com capacidade de perceber como é que os seus programas são executados, uma vez que nem sempre um programa sintaticamente correto exprime o utilizador quer fazer, tendo depois grandes dificuldades para determinar e compreender esses erros. Para tal, estes sistemas utilizam algumas abordagens como [KP05]:

1. **Programação Concreta: Atores em Micro mundos**— Os Micro mundos (do inglês Microworld), surgem como uma tentativa de tornar a programação mais concreta, uma vez que a maioria das outras linguagens introdutórias são linguagens abstratas. Por exemplo, a execução de operações matemáticas são feitas e guardadas utilizando registos invisíveis, o que dificulta aos iniciantes à programação a entender e a corrigir os erros nos seus programas. Estes sistemas permitem aos estudantes construir programas através do controle do comportamento de atores num pequeno mundo (baseado num mundo físico real). Os atores executam apenas um pequeno conjunto de ações o que facilita o domínio desses sistemas e das suas linguagens. Exemplo: ***Karel, Josef, Turingal***.

2. **Modelos de Execução de Programas** — Em vez da criação de uma linguagem que possui uma interpretação física, os sistemas nesta categoria fornecem um metáfora física para explicar as ações na maioria das linguagens. Os estudantes podem visualizar (ver simulações) a execução dos seus programas e talvez perceber a razão do porquê de algumas vezes estes não funcionam como eles esperariam. Exemplo: ***Toon Talk, Prototype 2,...*** etc.

Karel

Karel é um robô, Figura 3.9 vivendo num mundo representado num ecrã de um computador. Para controlar o robô o estudante terá que escrever pequenos programas com instruções para fazer movimentá-lo no mundo[Wil05].

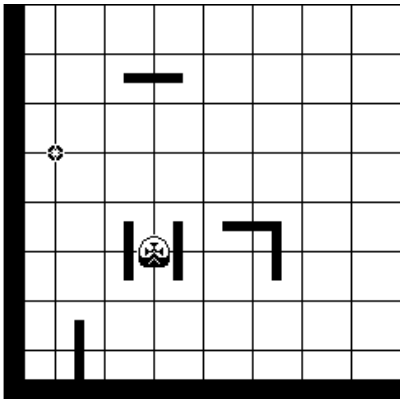

Figura 3.9: Interface do Mundo de Karel.

Toon Talk

O ToonTalk, Figura 3.10 é um jogo e uma linguagem de programação. Os objetos(cidades, criaturas, veículos) representam programas [KP05]. Estes programas podem ter diferentes finalidades: jogos, música, matemática, simulação, ciência, robótica ou educação.

As estratégias utilizadas para aprender Toon Talk passam por jogar os puzzles, ver exemplos, brincar e explorar. Existe também um Extra-terrestre *(ET)* que fornece ajuda quando há maior dificuldade em resolver determinados problemas [Kah00].

Figura 3.10: Ambiente Toon Talk.

3.1.4 Suporte ao Ensino

Os sistemas nas categorias anteriores, usam a simplificação dos mecanismos necessários a criação, estruturação e execução de um programa para uma maior facilitação da aprendizagem da programação para os iniciantes. Nesta categoria, os sistemas tentam facilitar o processo de aprendizagem da programação através da introdução dos conceitos de forma gradual e tornando possível a comunicação entre os estudantes aprendendo uns com os outros [KP05]. Exemplos: ***Algo Block, Pet Park, MOOSE Crossing, Cleogo***, ... etc.

MOOSE Crossing

É um ambiente de programação em rede construído para crianças. O ambiente é textual e uma adaptação Multi-User Dungeon(MUD). Os MUDs tem origens em jogos de aventuras textuais [Bru97].

A linguagem de programação, Figura 3.11, de Moose Crossing, é uma linguagem de script e orientada a objetos. Permite às crianças criar espaços e personagens, similares aos existentes nesses jogos, que habitam um mundo textual. Quando uma criança termina um projeto, qualquer outra criança no ambiente Moose pode interagir com esta.

Moose tem como público alvo crianças dos 8 aos 13 anos [Bru97] [KP05].

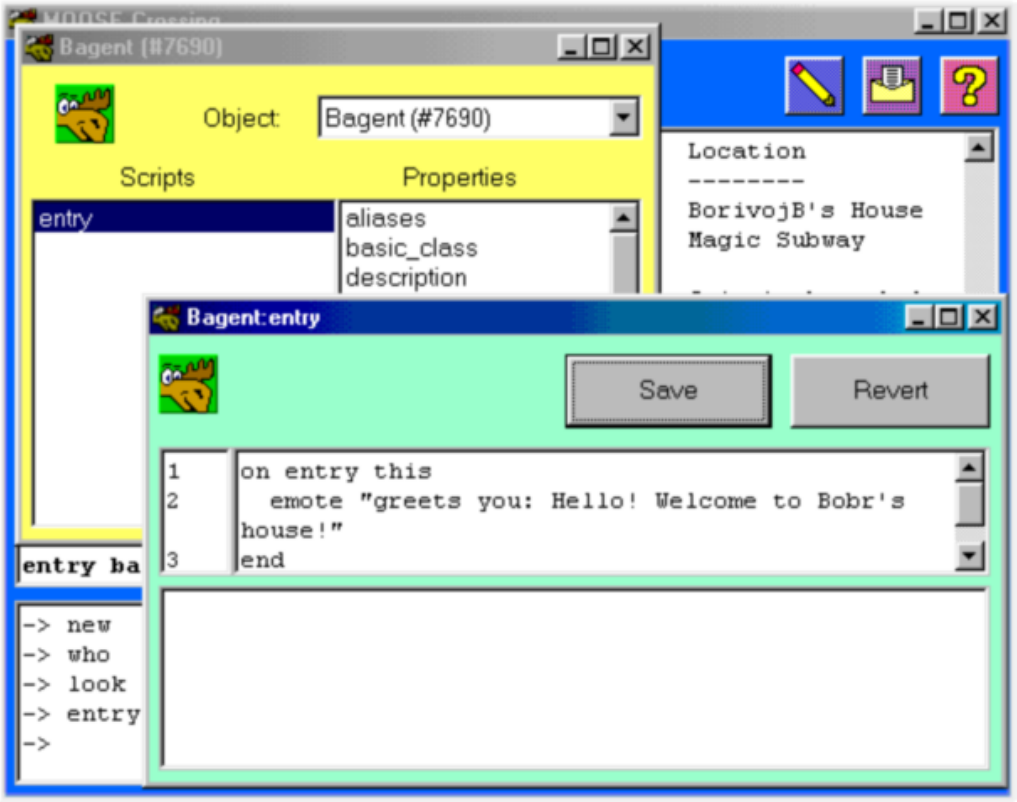

Figura 3.11: Interface do MOOSE Crossing.

Cleogo

É um ambiente de programação que permite à vários utilizadores criarem programas simultaneamente. É uma versão colaborativa do sistema Leogo(Single-user).

Cleogo fornece três formas alternativas,Figura 3.12 para criação de programas: demonstração (utilizando uma linguagem de manipulação direta),uma linguagem de ícones e uma linguagem tradicional de texto [CB98].

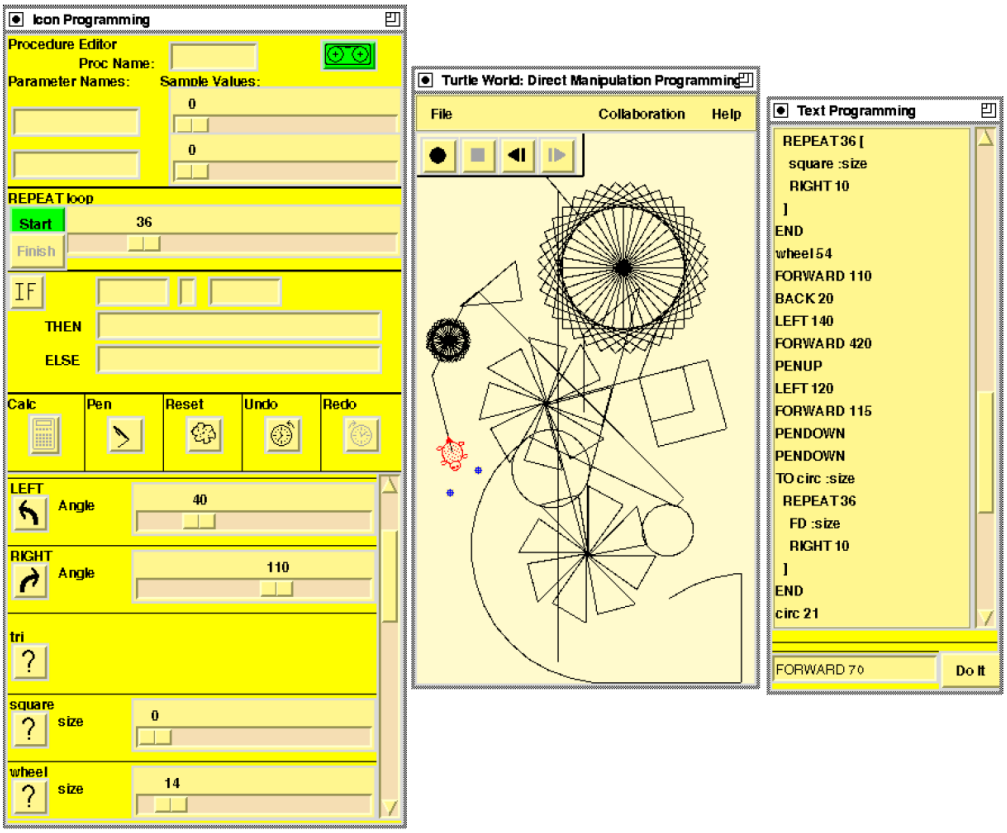

Figura 3.12: As três metáforas utilizadas por cleogo[CB98].

3.2 Programação como Suporte/Capacitação para outras áreas

As linguagens e métodos programação encontradas nos sistemas pertencentes a esta categoria tem como objetivo principal permitir às pessoas uma maior capacidade para criar coisas que satisfaçam as suas necessidades, quer como suporte a aprendizagem de outros conceitos, introdução dos mecanismos da programação ou na utilização da programação para promover outras áreas [KP05].

A Tabela 3.2 apresenta as abordagens utilizadas por estes sistemas, as suas caraterísticas e objetivos e alguns exemplos.

Tabela 3.2: Abordagens dos sistemas de suporte

Abordagens	Caraterísticas	Exemplos
Mecanismos de Programação	Exploram formas de melhorar as linguagens de programação e procurar formas alternativas de criar programas	*AgentSheets, Logo, SqueakEtoys, HANDS, Chart N Art, KidsRuby*
Promoção de Outras Áreas	Utilização da programação como ferramenta para promover outras áreas. Permitem maior controlo e criação e exploração de domínios particulares.	*Mindrover, Hank*

3.2.1 Logo

A linguagem de programação Logo é uma variante de *Lisp* com a maioria dos sinais de pontuação removidos de forma a torná-la mais acessível as crianças. O objetivo de Logo passa por permitir as crianças explorarem áreas como a matemática, música, robótica, telecomunicação e ciência.

Logo permite criar criar simulações e criar apresentações multimédia, e foi desenhada para ser bastante acessível mas também para suportar projetos mais sofisticados [Fou00].

A parte mais conhecida de Logo, é o ***Logo Turtle***(Figura 3.13), que é uma tartaruga robô que pode fazer desenhos no chão [KP05].

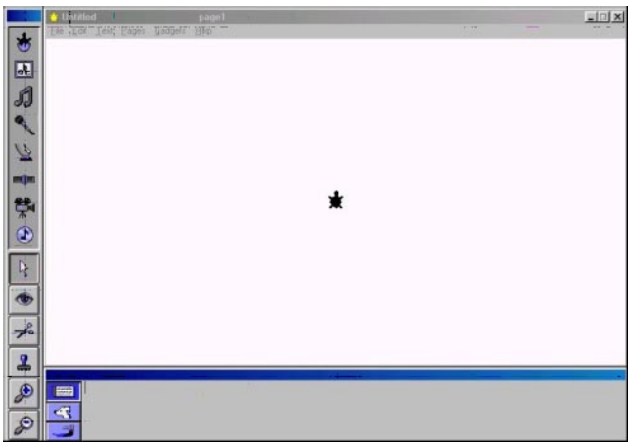

Figura 3.13: Interface Logo(Logo Turtle).

Logo inspirou o aparecimento de muitos outros softwares de investigação ou comercias, tais como [Fou11] [KP05]:

- **Comenius Logo** — Um versão de logo com fortes capacidades de animação.
- **MicroWorlds** — Um ambiente de exploração em áreas como a matemática, ciência, entre outras. Permite a criação de projetos Web e multimédia.

- **Terrapin Logo** — Desenhado para os gráficos tradicionais do logo turtle e a linguagem Logo.

- **StarLogo** — Um ambiente programável que permite aos estudantes explorar sistemas centralizados como colónias ou padrões de tráfego.

- **NetLogo** — Um ambiente de programação multi-agente. É inspirada no StarLogo.

A linguagem Logo é o percursor de todas as tentativas de tornar acessível a programação a iniciantes a programação, incluindo crianças.

3.2.2 AgentSheets

Em AgentSheets os estudantes têm a possibilidade de criar simulações baseadas em agentes e depois publicá-las na Web.

A interface do AgentSheets, Figura 3.14 é atrativa para o utilizadores, pois usa a filosofia *drag-and-drop*. Deste modo é fácil criar simulações interativas, explorar novas ideias, testar teorias e explorar processos complexos no campo da ciência [Age11].

As simulações são feitas especificando o comportamento dos *sriptes* 2D sobre um mundo representado por uma grelha. Os sprites podem ser movidos para novas posições, produzir sons ou mudar de aparência [KP05].

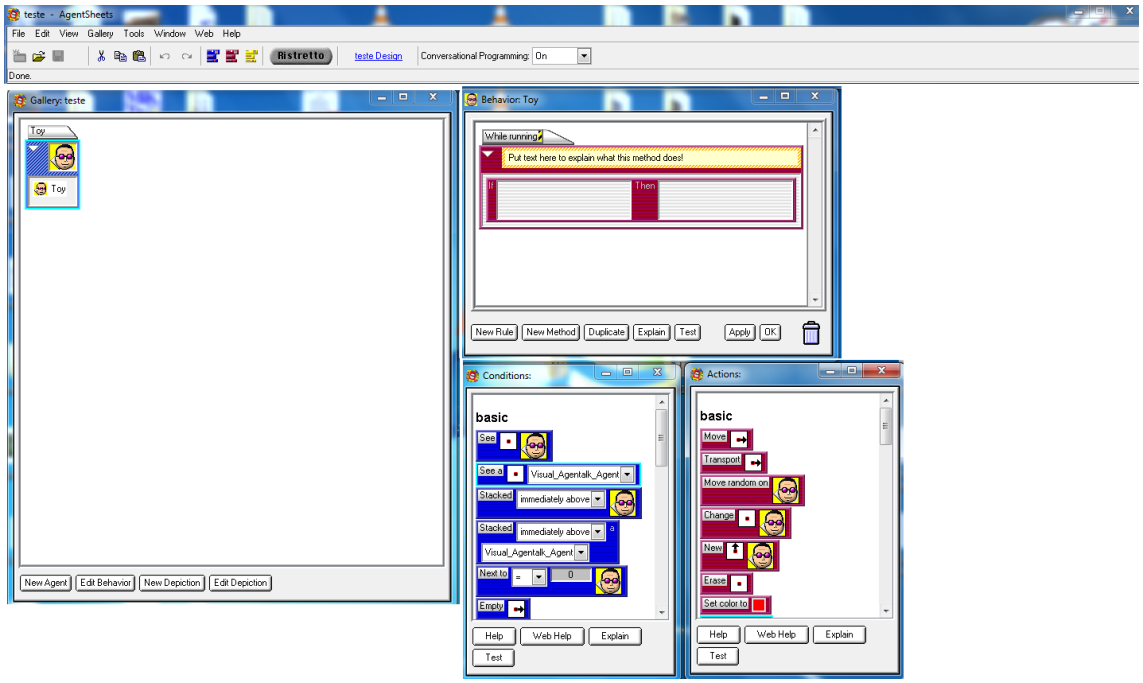

Figura 3.14: Interface do AgentSheets.

3.2.3 Squeak Etoys

É um ambiente, ver Figura reffig:etoys, rico para criação de conteúdos media com um modelo de objetos baseado em scripts. Possui modelos para vários tipos de objetos criado pelos utilizadores e que correm em várias plataformas. Squeak Etoys suporta gráficos 2D e 3D, imagens, texto, partículas, apresentações, páginas Web, vídeos e sons, ... etc.

Uma das caraterísticas do Etoys é a partilha em tempo real dos projetos criados com outros utilizadores do sistema [KK05].

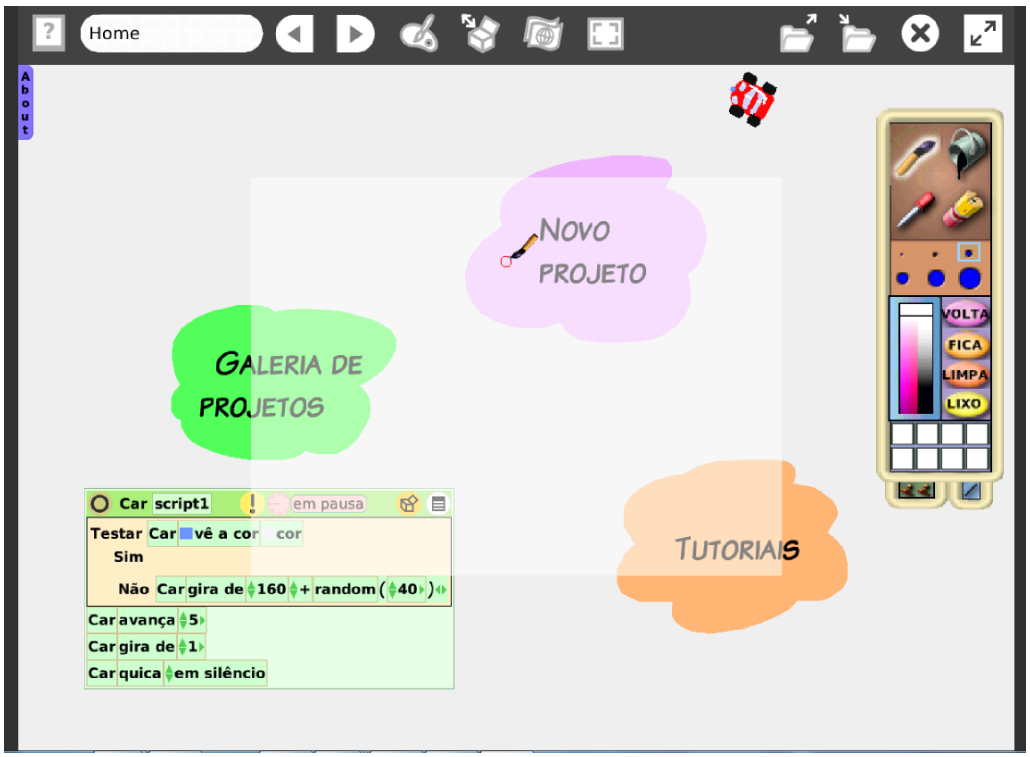

Figura 3.15: Interface do Etoys.

3.3 Resumo

São muitas as linguagens e ambientes de programação quer surgiram juntamente com os computadores pessoais tentando tornar mais acessível a programação a iniciantes ou mesmo introduzir os conceitos da programação num nível ainda mais baixo, à Crianças.

A linguagem Logo criada por Seymour Papert é o grande percursor de todas as restantes linguagens e ambientes de programação que vem surgindo desde então.

Todos estes sistemas, tem como objetivo tentar mostrar aos seus utilizadores as vantagens de aprender programação e ao mesmo tempo dar lhe possibilidade de criarem coisas que satisfaçam as suas necessidades, embora as abordagens/metodologias utilizadas em cada um deles possa ser diferente. Uns preocupam-se mais com o ensino dos mecanismos da programação de forma mais simplificada possível, e outros procuram disponibilizar ferramentas que usam a programação para incentivar e motivar as crianças em outras áreas.

De entre todas as linguagens e ambientes de programação apresentados, os que chamaram mais à atenção foram o *Scratch*, *Squeak Etoys*, *Alice* e *Kodu*, pois estes são altamente visuais, o que facilita a interação, e o conceito de blocos de instrução em vez do tradicional texto é uma mais valia para as crianças, uma vez que estes conseguem criar os programas de forma fácil e intuitiva, encaixando os blocos uns nos outros.

Estes pontos fortes vão ser utilizados nesta dissertação, ao mesmo tempo que se cria algo de novo que sirva como ponto de ligação com os jogos.

Capítulo 4

Boobo World - Game Design

4.1 Conceito

Boobo World é um MMOG onde os jogadores são convidados a explorar um mundo virtual através de um avatar e do seu robô, o Boobo. O Boobo é um robô programável que, inicialmente, o único comportamento que possui é seguir o seu dono para todos os sítios onde este vá. Cabe a cada jogador programar o seu robô com novos comportamentos, tornando o mais esperto e independente.

Boobo world é um jogo destinado a ser jogado online, podendo assim ser acessível a uma maior gama de utilizadores. Este jogo é também uma rede social, suportando funcionalidades como chat, árvore de amigos, mensagens privadas, partilha e concursos públicos. Este jogo tem como público alvo crianças dos 8 aos 13 anos de idade.

4.2 Objetivos

Boobo World como jogo sério tem como objetivo principal a transmissão de conceitos da programação aos jogadores. Porém, este objetivo diz respeito ao criador do jogo, um jogador terá de cumprir objetivos mais específicos ao jogar. Estes objetivos são:

- **Criar e Personalizar o seu personagem** - Este é o primeiro grande objetivo do jogo, em que cada jogador tem que criar e personalizar o seu avatar e depois adotar um robô que o acompanhará no resto do jogo. Este robô, o Boobo, é também personalizável.

- **Resolução dos desafios** - A progressão no jogo é conseguida através da resolução com sucesso dos desafios de programação. Esta resolução pode ser realizada individualmente ou em modo colaborativo.

- **Rede Social**- Todos os jogadores vão ter como objetivo criar uma rede de amigos com o qual podem interagir no ambiente de jogo, utilizando o chat, mensagens privadas, partilha, convite para resolução dos desafios de programação, etc...

- **Concursos** - Durante o jogo vão surgindo concursos no qual os jogadores podem participar e assim ganhar mais recompensas, podendo mesmo avançar nos níveis.

Estes objetivos serão integrados no jogo, atuando diretamente com toda a mecânica deste.

4.3 Linguagem e Ambiente Programação

Boobo World é um jogo sério e, como tal, existe uma componente pedagógica a ser transmitida aos jogadores, neste caso, conceitos básicos de programação. Estes conceitos serão passados através dos desafios de programação presentes no jogo e da linguagem de programação que as crianças utilizarão para resolver estes mesmos desafios.

A programação do Boobo, de forma a este superar os desafios, é feito numa nova linguagem de programação baseada no Kodu e adaptada ao contexto do jogo.

A seguir é apresentado uma descrição das duas linguagens utilizando a notação das gramáticas livres de contexto. Nesta notação uma descrição de uma linguagem é representado por uma série de regras de produção, onde o lado esquerdo(LHS) mostra uma variável não terminal e o lado direito (RHS) contém as variáveis terminais e não terminais. Cada variável terminal é um elemento do alfabeto da linguagem. As variáveis não terminais começam sempre por uma letra maiúscula e as terminais por minúsculas [Sto10].

- **Linguagem Kodu(versão básica)**

  Game → Actors
  Actors → Object| Object Actors
  Object→ Page Object| Page
  Page Rule Page | Rule
  Rule → Condition Action | Condition Action Page
  Condition → Sensor FilterSet |
  Action → Actuator Selector ModifierSet | Actuator ModifierSet |
  ModifierSet → Modifier ModifierSet | Modifier
  FilterSet → Filter FilterSet | Filter
  Sensor → see | hear | bump | ...
  Filter → apple | blue | health | ... | ε
  Actuator → move | shoot | add | ...
  Selector → toward | me | avoid | ... | ε
  Modifier → 5 points | red | quickly | ... | ε

Em Kodu a variável de ínicio é o *Game*, e contém a variável *Actors* que é um conjunto de *Objetcs*. A programação de cada *Object* é definida pelo menos por uma *Page*, e cada *Page* tem uma ou mais regras.

Uma regra(*Rule*) é uma linha de programação na forma de uma condição(*Condition*) e de uma acão(*Action*), por exemplo " *when see apple red, do move toward quickly* ", onde a primeira parte da frase é a condição e a segunda a ação.

A derivação deste exemplo, utilizando a gramática apresentada, é obtida fazendo uma árvore de análise em vez da substituição das regras. Esta árvore é conhecida como AST (*Abstract Syntax Tree*) [Sto10].

- **Linguagem Programação no Boobo World**
  Esta nova linguagem utiliza a mesma gramática apresentada anteriormente(gramática básica do Kodu), as diferenças encontram-se nas variáveis terminais, ou seja o conjunto composto pelos *Sensors*, *Filter*, *Actuator*, *Selector* e *Modifier*.

  Os elementos que compõe este conjunto representam os blocos de instruções disponíveis no ambiente de programação e são apresentados a seguir:

  Sensor $\rightarrow$ see | health | bump | click| ε
  Filter $\rightarrow$ wall | bot | key | ramp | charger | door | light | water | ε
  Actuator $\rightarrow$ eat | move | pick | say | stop | turn | jump | ε
  Selector $\rightarrow$ foward | right | left | back | hello | bye | 5 energie | 10 energie | 50 energie | 100 energie | ε
  Modifier $\rightarrow$ fast | slow | ε

Os sensores (Sensor) permitem determinar condições do ambiente ou estado do robô. São utilizados em conjunto com os filtros(Filter), objetos do mundo virtual, na construção de uma condição.

As ações (Actuator) representam as ações básicas que o robô pode efectuar no mundo. Estas podem, ou não, ser acompanhadas com instruções do conjunto *Selector* e *Modifier*.

Ao combinar estes elementos no ambiente de programação, o jogador estará a criar uma nova regra para o seu robô. Esta combinação respeita a relação que existe entre as instruções, por exemplo quando um jogador escolhe a instrução *see*, a instrução do conjunto *Filter* que virá a seguir terá de fazer sentido.

Estando a linguagem de programação definida, o próximo passo será a implementação de um ambiente de programação visual, ver capítulo 5, atrativo e de fácil interação.

A Figura 4.1 mostra um esboço deste ambiente de programação e os seus principais componentes.

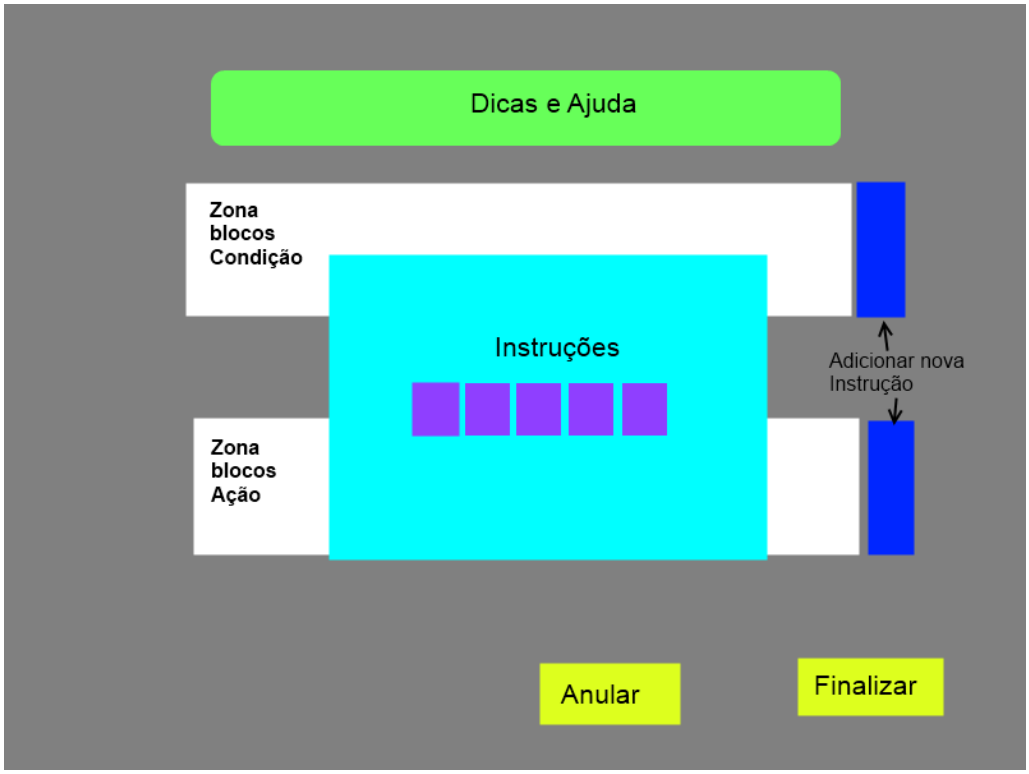

Figura 4.1: Esboço ambiente programação no Boobo World.

Este ambiente será constituído pelos seguintes elementos:

- **Dicas e Ajuda:** Ao entrar no ambiente de programação, o utilizador conseguirá ver uma área com mensagen explicativas do funcionamento do ambiente.
- **Zona blocos Condição :** Área destinado a colocação dos blocos de instrução que formam uma condição.
- **Zona blocos Ação :** Todos os blocos de instrução que podem fazer parte de uma ação, e caso sejam escolhidos são colocados nesta área.
- **Instruções :** Correspondem as instruções definidas na linguagem apresentada anteriormente. Além do texto, estas instruções possuirão uma imagem, no mesmo contexto, que ajudarão o utilizador no processo de construção da regra.
- **Botões de Anular e Finalizar:** O botão de anular poderá ser utilizado sempre que o utilizador pretende anular a última ação, última instrução escolhida, efetuada no ambiente.

Enquanto que o botão finalizar permitirá ao utilizador deixar o ambiente de programação mesmo que ele não tenha criado uma regra.

O ambiente de programação esta oculto no jogo, a sua visualização acontece sempre que um utilizador acede a um desafio e aceita resolvê-lo.

4.4 Mecânica

A mecânica do Boobo World pretende ser fácil e intuitiva, tendo em conta o seu público alvo. No diagrama de atividades, Figura 4.2, pode-se ver os principais estados que um jogador pode percorrer durante o jogo.

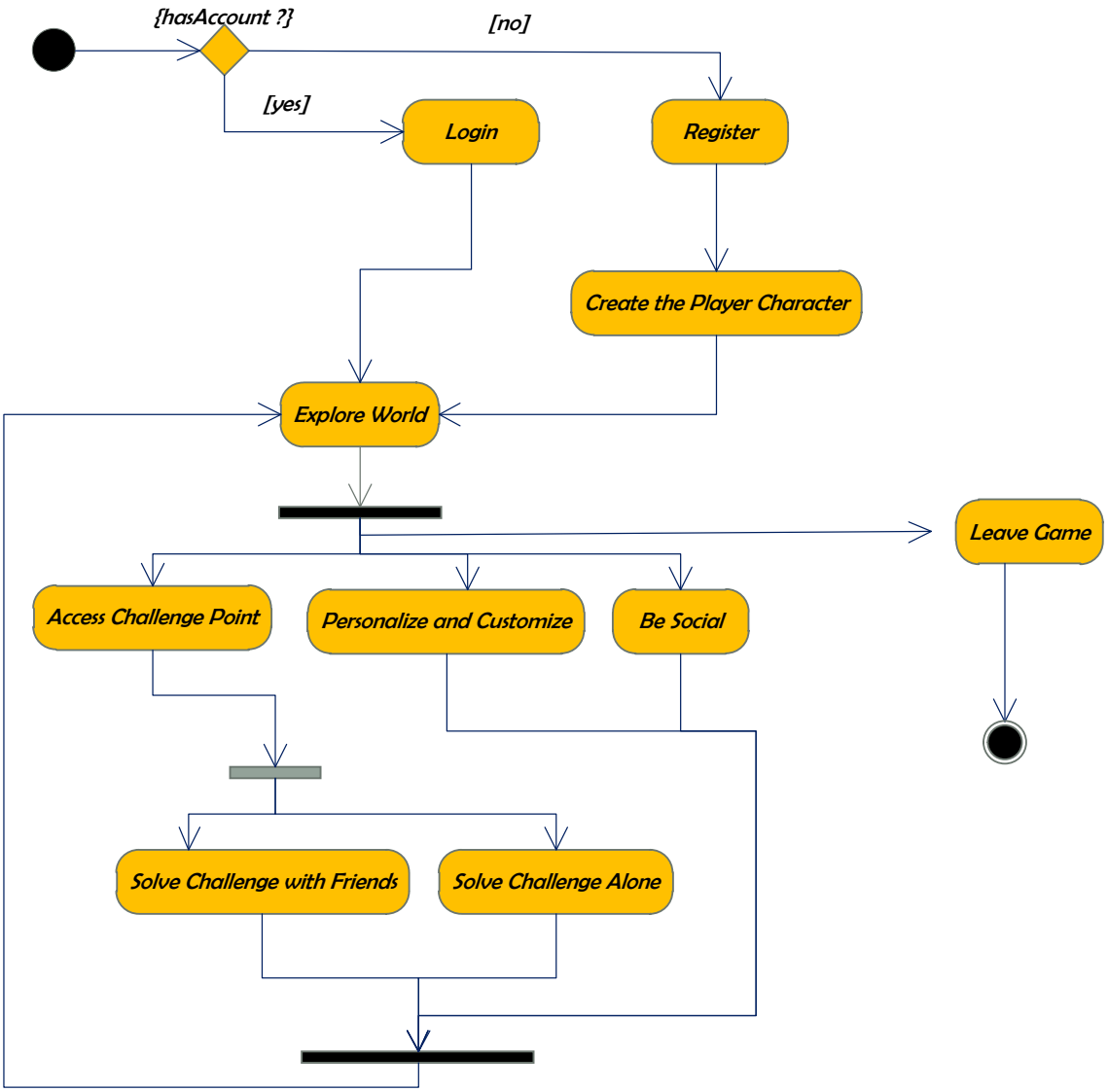

Figura 4.2: Diagrama atividades- Mecânica do Boobo World

Se o jogador já estiver registado no jogo, este entra no jogo(*Login*) e depois será encaminhado diretamente para o mundo virtual(*Explore World*), caso contrário é necessário fazer o registo do jogador(*Register*), criar o seu personagem(*Create the Player Character*), e só depois de ter o seu personagem é que o jogador poderá juntar-se aos restantes no mundo.

Estando no mundo virtual, o jogador pode optar por fazer algo mais específico como, aceder aos desafios através dos respetivos pontos de acesso(*Access Challenge Point*), ir para sua área pessoal e proceder a personalização e customização do seu personagem(*Personalize and Customize*), ou então interagir com os outros jogadores (*Be Social*). Estando neste dois últimos estados, há um conjunto de outros estados que vão ser desencadeados no jogo e para o qual o jogador poderá seguir.

Ao aceder a um desafio, o jogador poderá sempre optar por resolver os desafios sozinho(*Solve Challenge with Friends*) ou então em conjunto com mais jogadores(*Solve Challenge Alone*). Após a resolução dos desafios o jogador regressa ao mundo virtual e o processo começa de novo.

Ainda no mundo virtual, o jogador pode optar por abandonar o jogo, não perdendo dessa forma o seu progresso no jogo.

4.5 Níveis

Um nível em Boobo World é definido pelo desafio de programação que o jogador terá de resolver, isto é, um nível corresponde a um desafio de programação. Sempre que o jogador acede a um desafio e o resolva com sucesso, este passa para um próximo nível.

Todos os desafios de programação em Boobo World podem ser realizados por apenas um ou vários jogadores ao mesmo tempo. O acesso a estes desafios faz-se através dos pontos de acesso existentes no mundo virtual, objetos do mundo com o qual o jogador pode interagir. Cada um destes objetos é uma "porta" para aceder a um determinado desafio e este é único em cada nível.

A medida que o jogador avança no jogo, passando de nível os desafios de programação vão-se tornando mais complexos. Assim, os desafios de programação em Boobo World podem ser divididos em dois grandes grupos:

1. **Desafios Simples** - Estes correspondem a desafios em que os jogadores apenas tem de construir uma função, de forma a completar um determinado objetivo.

2. **Desafios Compostos** - Os desafios compostos pretendem incentivar os jogadores a reutilizar as funções criadas nos níveis anteriores em conjunto com outras novas que terão de criar.

Na tabela 4.1 podemos ver um conjunto de desafios simples projetados para o Boobo World.

Tabela 4.1: Desafios simples no Boobo World.

ID	Descrição	Solução
1	Ensina o teu bot a dizer qualquer coisa quando clicas nele.	*When click bot Do say hello* ou*When click bot Do say bye*
2	Boobo esta a dirigir-se contra uma parede e não sabe o que fazer. Ensina-o a parar.	*When see wall Do stop*
3	Boobo distraiu-se e acabou por perder-se numa ruela sem saída, ajude-o a voltar ao ponto de partida.	*When see wall Do move back*
4	O teu bot anda a procura de uma chave numa sala cheia de outros objetos, ajuda-o a encontrar a chave certa e apanhá-la	*When see key Do pick key*
5	Ensina o teu bot a pegar num carregador.	*When see charger Do pick charger*
6	Ensina o Boobo a virar-se para direita quando vê uma parede.	*When see wall Do move rigth*
7	Ensina o Boobo a virar-se para esquerda quando vê uma parede.	*When see wall Do move left*
8	Ensina o Boobo a subir uma rampa.	*When see ramp Do move forward.*
9	Ensina o Boobo a evitar uma poça de água	*When see water Do stop* ou *When see water Do move back* ou *When see water Do move rigth* ou *When see water Do move left*
10	Ensina o Boobo a carregar a sua bateroa	*When health empty Do move eat x energie*. Onde x pode ser 5, 10, 50,100.

Em cada um dos desafios temos um identificador (ID), uma descrição e a solução do desafio. Esta solução é uma regra na forma de uma condição e de uma ação e para alguns desafios existe mais do que uma possibilidade.

Estes dez desafios são exemplos dos vários desafios que podemos projetar para o Boobo World, tendo em conta as instruções definidas na linguagem.

4.6 Avatar e Robô

O avatar é a representação do jogador no jogo. Esta representação pode incluir todos os elementos que o jogador quiser e que o jogo permita fazer. Sendo este jogo destinado as crianças, todos os avatares terão a forma de crianças, de ambos os géneros, e cada jogador poderá personalizar e customizar o seu avatar de acordo com as sua preferência.

O robô, chamado de Boobo é o elemento do jogo que pretende criar uma ligação emocional com os jogadores. Pois, um Boobo é um robô completamente dependente do seu dono e à medida que o jogador vai ensinando-lhe novos comportamentos novos laços serão criados.

O avatar do jogador e o seu Boobo são personalizados de forma independente, no entanto, no mundo virtual há um comportamento muito especial do Boobo que é seguir o avatar para todo o lado onde este vá, exceto nas situações dos desafios de programação, Figura 4.3.

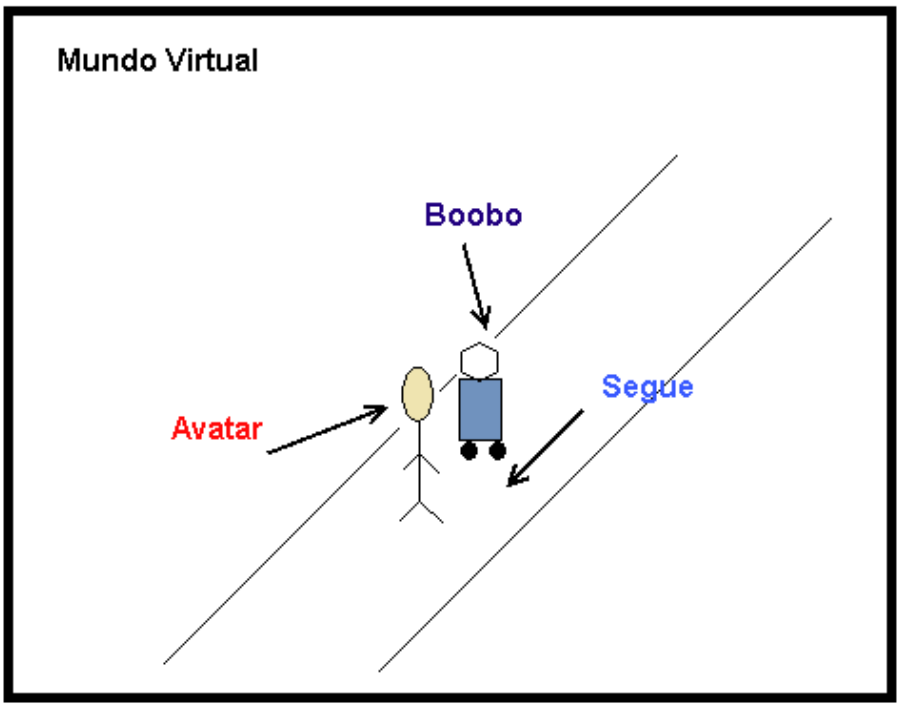

Figura 4.3: Comportamento especial do Boobo no Mundo Virtual

Assim o avatar e o seu respetivo Boobo constituem os elementos do jogador no jogo.

4.7 Interface do Utilizador

Durante o jogo existe um conjunto de informações disponíveis aos jogadores. Esta informação esta disponível em vários locais e sobre diversas formas e pretende facilitar a interação no jogo. Em seguida é apresentada a Tabela 4.2 que lista essas informações, a sua descrição e a sua localização no jogo.

Tabela 4.2: Informações presentes no Boobo World.

Informação	**Descrição**	**Localização**
Painel de Controlo	Durante o jogo, os jogadores terão disponível um painel de controlo com as seguintes funcionalidades: um botão para aceder à área pessoal do jogador, um botão para ir às lojas, um botão para iniciar o chat e um botão para aceder às mensagens privadas.	Mundo Virtual
Mapa	Todos os jogadores poderão escolher um sítio para ondem querem ir no mundo virtual, através de um mapa interativo. Este mapa é acedido através de um botão sempre visível no ecrã do jogo.	Mundo Virtual
Status do Jogador	O nome do jogador e a sua pontuação, nível no jogo podem ser vistos por todos, bastando para isso carregar no avatar do respetivo jogador	Mundo Virtual
Enviar Convite a amigos	Durante a resolução de um desafio, o jogador terá a possibilidade de convidar outros jogadores para colaborarem. Este convite é feito através de um botão especialmente concebido para esse efeito.	Área Desafio
Programar o Boobo	Um botão dará acesso ao ambiente de programação onde o jogador pode criar as regras para a resolução do desafio.	Área Desafio
Executar o Desafio	Executar o desafio para verificar se a solução dada é a correta	Área Desafio
Lista de soluções	Quando os desafios são resolvidos em modo colaborativo, as diferentes soluções dadas pelos jogadores são visíveis para todos até o líder fazer uma escolha.	Área Desafio
Solução selecionada	Em modo colaborativo, quando o líder escolhe uma solução esta é a mesma para todos os jogadores, e será com essa que todos vão executar o desafio	Área Desafio

Para além destas informações, o jogador poderá requer dicas e ajuda sempre que o desejar.

4.8 Controlos

A interação no Boobo world será feita através do teclado e do rato. Podem ser usados em conjunto ou individualmente, dependendo das situações.

Com o teclado, ver Figura 4.4, o jogador pode inserir a sua informação de login, registar no sistema, conversar no chat, escrever mensagens privadas e fazer zoom no mundo virtual utilizando as teclas Q(zoom in) e Z(zoom out).

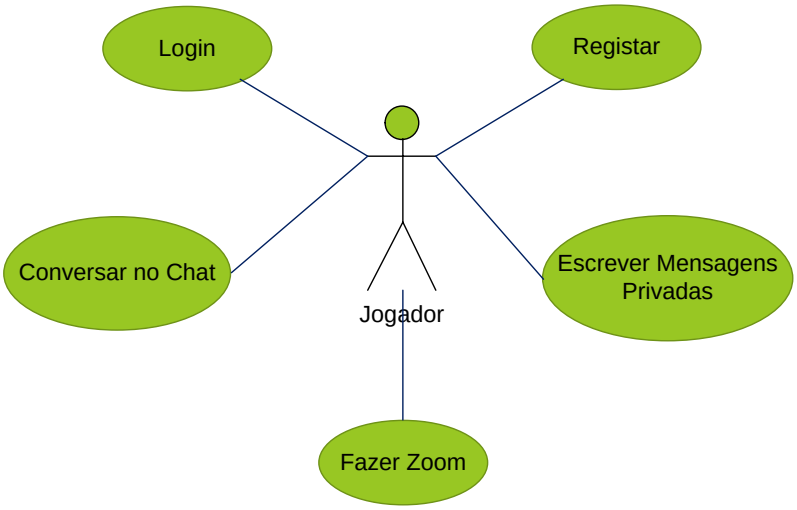

Figura 4.4: Diagrama Casos de uso com o teclado.

Utilizando o rato, ver Figura 4.5, cada jogador tem a possibilidade de mover o seu avatar, clicando numa nova posição no mundo virtual, entrar e sair dos desafios de programação, selecionar os blocos de instrução para criar uma regra ou interagir com outros botões (outras funcionalidades) no jogo.

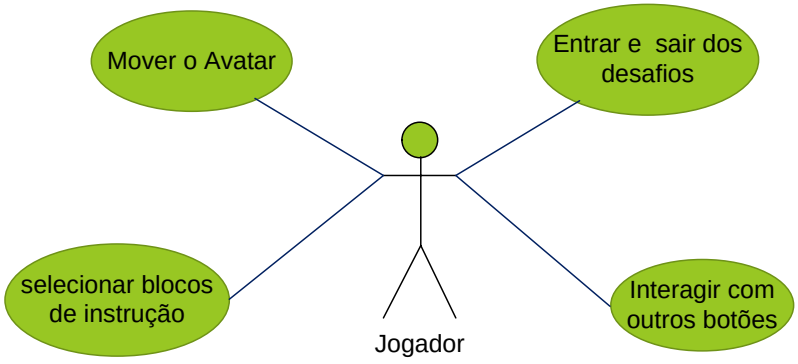

Figura 4.5: Diagrama casos de uso com o rato.

4.9 Arquitetura

A arquitetura do Boobo World é cliente/servidor. Esta arquitetura permite que o jogo seja acedido por vários jogadores(clientes) ao mesmo tempo e que haja comunicação entre eles, Figura 4.6.

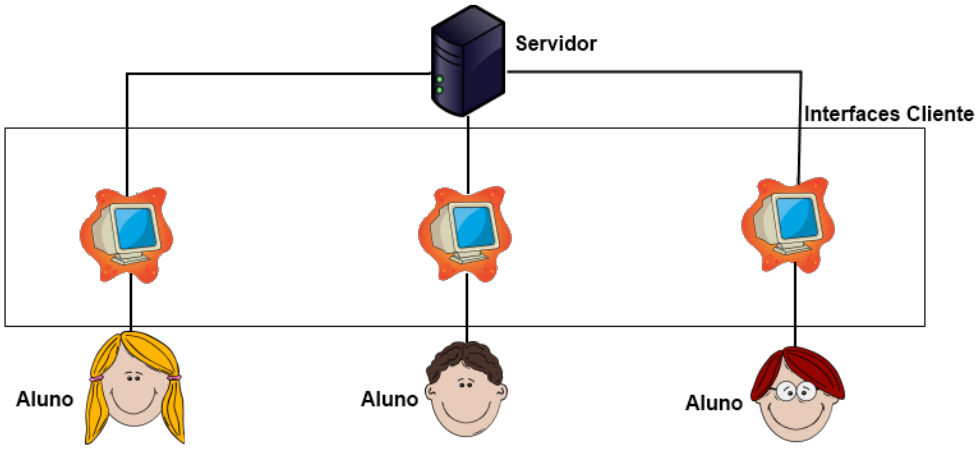

Figura 4.6: Arquitetura cliente/servidor

Cada cliente controla o seu personagem numa interface do jogo(mundo virtual) e executa um conjunto de ações que posteriormente são enviadas para o servidor onde o jogo se encontra alojado. O servidor recebe estas ações, aplica as alterações ao mundo e depois notifica os restantes clientes das modificações ocorridas. O servidor é também responsável pela notificação,dos outros jogadores, sobre entrada e saída de um determinado jogador.

Outros aspetos a ter em consideração na estruturação de um jogo em rede, tem a ver com a conexão do cliente ao servidor, a privacidade de cada um e a forma como a informação é transmitida. Existe duas abordagens comprovadas para esse efeito [uni11]:

- **Servidor Autoritário** Com esta abordagem o servidor fica responsável por fazer toda simulação do mundo, aplicação das regras do jogo e processamento dos *inputs* dos jogadores.

- **Servidor Não Autoritário** O servidor não controla o resultado dos *inputs* dos jogadores. Cabe a cada cliente fazer esse processamento e a lógica do jogo, e depois enviar o resultado de determinadas ações para o servidor. Este por sua vez trata de fazer a sincronização das ações no estado do mundo.

Para implementação do protótipo do Boobo World foi escolhida a segunda abordagem, Servidor não autoritário, pois apresenta maior facilidade de implementação, o servidor atua somente como intermediário para troca de mensagens entre os clientes, e o fato de a primeira abordagem exigir mais recursos de hardware.

4.10 Tecnologias e Ferramentas

As tecnologias atualmente mais utilizadas na criação de jogos para web são: **Flash**, **HTML5** e **Unity**. Todas elas apresentam vantagens e desvantagens em algumas situações específicas. A seguir é apresentada uma análise das três tecnologias, a partir dos seguintes critérios [Ric11] :

- Workflow - Análise da facilidade de criar jogos atualmente.
- Estabilidade - Análise da possibilidade de haver alterações desde o início da criação do jogo até a sua finalização.
- Distribuição - Análise da facilidade de disponibilizar o jogo aos jogadores.
- Segurança - Análise da proteção do código e do endereço IP.
- Licenciamento - Análise do custo da utilização da tecnologia para desenvolver o jogo.

Para cada um dos critérios apresentados, a análise das três tecnologias é a seguinte:

1. **Workflow**
   Flash: Criar jogos em flash é fácil. Existem muitos livros, sites, tutoriais, frameworks, livrarias, videos e exemplos de código disponíveis.

   HTML5: Criar jogos em HTML5 é complexo, no entanto existem alguns editores que pretendem facilitar o desenvolvimento.

   Unity: Tal como em flash, existe muita documentação disponível que ajuda o utilizador a criar um jogo em pouco tempo. O Unity fornece um editor visual muito fácil de utilizar e extremamente útil.

2. **Estabilidade**
   Flash: A plataforma onde o jogo "vive" é independente do sistema operativo, exceto iOS, e do tipo de broswser utilizado. Não faz diferença se o jogo esta a correr numa versão antiga do Internet Explorer ou na última versão do Chrome, a experiência será a mesma.

   HTML5: A tecnologia e as plataformas estão em constante mudança. Com isto podem acontecer dois efeitos secundários: O primeiro é que um jogo que esteja a correr sem problemas numa versão do broswer, poderá deixar de funcionar na versão seguinte do mesmo. O seguinte tem haver com a necessidade de pensar os jogos para todos os browsers existentes , o que poderá não ser uma tarefa fácil.

   Unity: O Unity possui um plugin próprio para correr os jogos na web, *Unity Web Plugin*. Este apresenta os mesmo benifícios que o flash.

3. **Distribuição**

   Flash: Um jogo desenvolvido em flash é arquivado num único ficheiro. Este ficheiro pode ser simplesmente servido a partir de qualquer site que queira hospedá-lo, ou então pode ser acedido através de um URL específico de uma máquina local.

   HTML5: Ainda não existe uma forma consistente de arquivar um jogo feito nesta tecnologia que funcione em todos os browsers. Deste modo a sua distribuição torna-se um bocado mais complexa quando comparada com os jogos em flash ou Unity.

   Unity: A distribuição de um jogo desenvolvido em Unity é feita de forma semelhante à da tecnologia flash.

4. **Segurança**

   O único ficheiro produzido pelo flash e Unity, não são perfeitos, porém combinam todos os elementos e código do jogo num único ficheiro que requer um know-how técnico considerável ou ferramentas de terceiros para poder ser corrompido. Em HTML5 o código pode ser ofuscado mas não encriptado com algo que depois não possa ser desencriptado por parte do cliente. Os elementos gráficos e de áudio são extremamente fáceis de copiar.

5. **Licenciamento**

   Tanto o flash como o HTML5 são grátis, no entanto há um conjunto de ferramentas comercias que podem ser utilizadas para melhorar e facilitar a criação do jogo. O Unity é grátis sem ser a versão *Pro*, e com este é possível criar um jogo completo.

Tendo em conta a análise desses critérios selecionou-se o Unity. Para além dos seus jogos correrem na web, estes também podem ser disponibilizados para outras plataformas como PC, Mac, IOS e todas as consolas de jogos atuais, Wii, Xbox 360 e PlayStation 3 [uni12].

No próximo capítulo é apresentado a implementação de um protótipo do Boobo World utilizando a tecnologia Unity.

Capítulo 5

Boobo World - Implementação

Criar um Jogo é um processo contínuo e longo, que envolve várias equipas(designers, programadores, marketing) a trabalharem em conjunto. E como tal, no âmbito desta dissertação procedeu-se à implementação de um protótipo funcional que fosse capaz de avaliar a fiabilidade do Boobo World no que diz respeito a introdução dos conceitos de programação de forma colaborativa.

Nesta secção são apresentadas as opções de implementação do protótipo Boobo World.

5.1 Mundo Virtual

Um mundo virtual é composto por um conjunto de elementos visuais com o qual o jogador pode interagir, este protótipo apresenta três elementos essenciais que podem ser observados na Figura 5.1:

1. **Terreno** É a representação de um espaço físico no mundo. Este elemento é o primeiro a ser visualizado pelos jogadores quando entram no jogo, ainda antes do processo de *login*.

2. **Avatar do Jogador e Boobo** Estes dois elementos constituem o personagem do jogador no jogo. O Boobo é o robô que esta sempre atrás do seu dono, o avatar do jogador.

3. **Pontos Acesso Desafio** Este é o elemento responsável pela ligação entre o mundo virtual com os desafios e o ambiente de programação. A interação entre o personagem do jogador com este elemento, contato, permite visualizar qual o respetivo desafio e o jogador pode optar por resolvê-lo ou não.

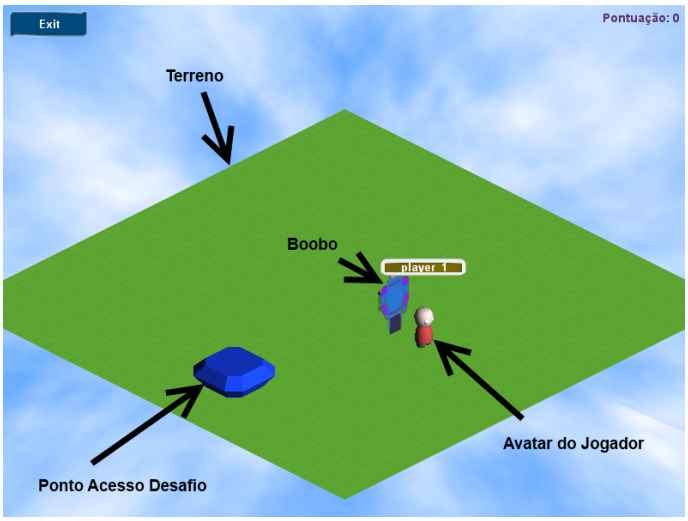

Figura 5.1: Terreno do mundo.

5.2 Níveis

O protótipo Boobo World é constituído por três níveis, cada um correspondendo a um desafio de programação. No primeiro nível, Figura 5.2, os jogadores têm de ensinar o seu boobo a falar.

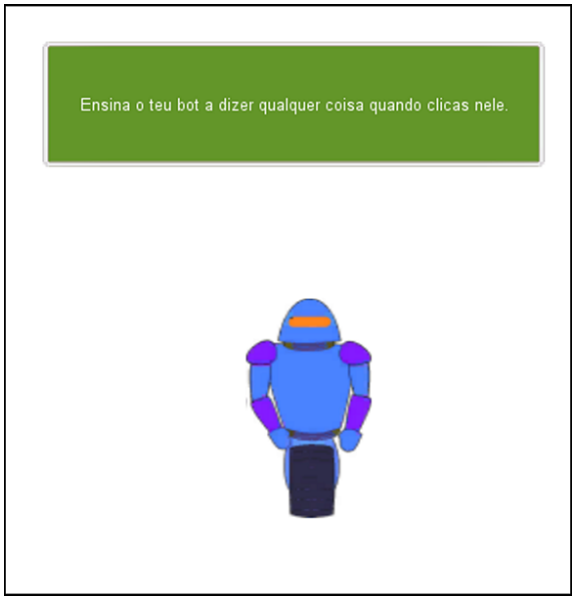

Figura 5.2: Nível 1 - Ensinar o boobo a falar.

No nível 2, Figura 5.3, os jogadores terão como objetivo ensinar o boobo a parar quando este vê uma parede.

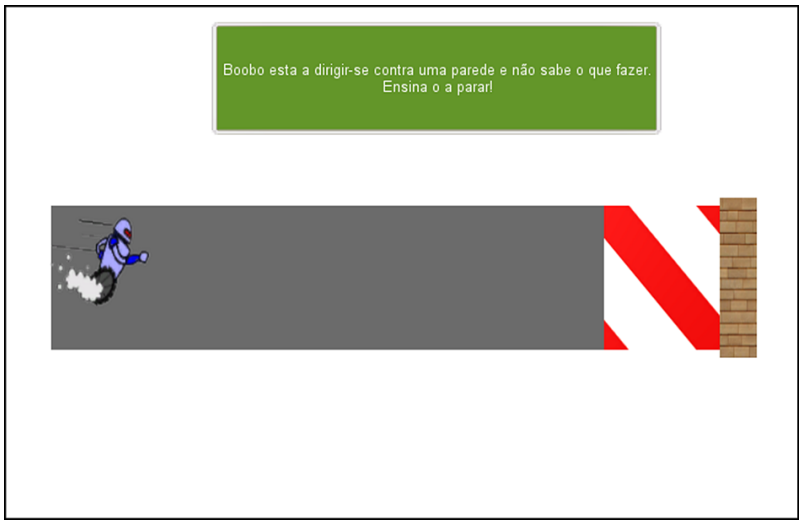

Figura 5.3: Nível 2 - Ensinar o boobo a parar.

Por fim, no nível 3, Figura 5.4, quando o boobo vê uma parede já sabe parar, e neste desafio os jogadores terão de criar uma regra para o fazer voltar ao ponto de partida.

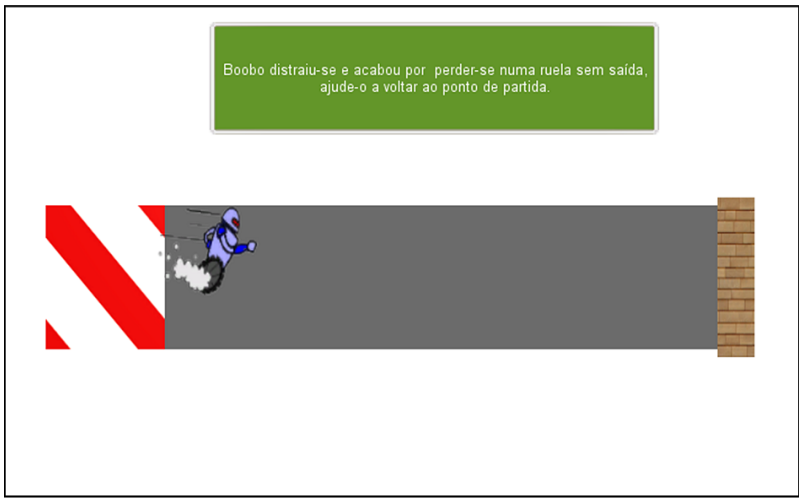

Figura 5.4: Nível 3 - Ensinar o boobo a voltar ao ponto de partida.

Em todos os desafios os jogadores terão a possibilidade de colaborar uns com os outros. Este processo será apresentado na próxima secção.

5.3 Desafios e Colaboração

Boobo World tem como um dos objetivos cativar os jogadores para resolução dos desafios de programação de forma colaborativa.

Cada desafio tem uma descrição, uma cena, e uma solução. A solução corresponde a uma regra que os jogadores terão de implementar, e esta tem de ser igual ao que foi definida na criação do desafio, ver Figura 5.5.

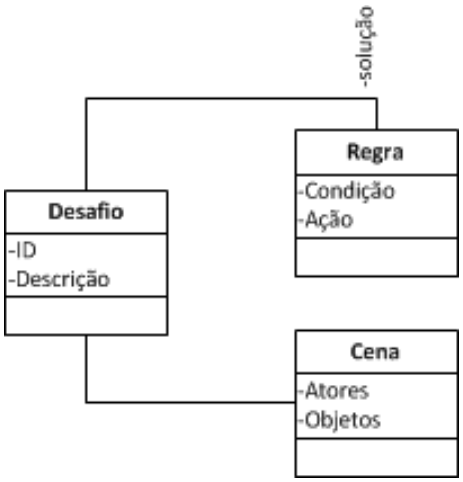

Figura 5.5: Um desafio no Boobo World.

Enquanto o jogador não acerte na solução do desafio, este permanecerá no mesmo desafio e pode convidar outros jogadores para resolução do mesmo. Este convite é dirigido apenas aos jogadores que estejam no mesmo nível do jogo, isto é, que ainda não tenham resolvido esse desafio. Na Figura 5.6 pode-se ver como este processo se desenrola. Primeiro o jogador *Jogador1* envia um convite para resolução do desafio *desafio 1* e apenas o *Jogador3* e *Jogador4* recebem o convite, uma vez que o jogador *jogador2* esta noutro nível.

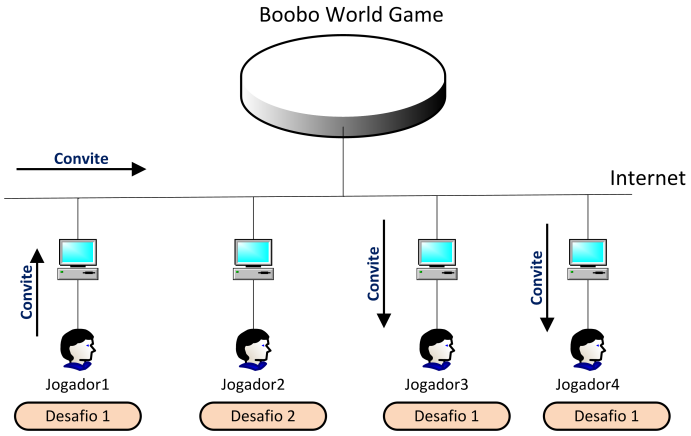

Figura 5.6: Convidar outros jogadores para resolver um desafio.

Após o envio do convite para os outros jogadores inicia-se todo um processo de colaboração se o convite for aceite por todos ou qualquer um dos outros jogadores. Este processo esta descrito no diagrama de sequência, Figura 5.7, onde *Jogador1* é nomeado o líder na primeira ronda, enviou o convite, e todos os outros jogadores são colaboradores.

A partir do momento em que todos os jogadores estejam conectados no mesmo desafio, cada um faz a resolução individual do desafio, a solução.Depois cabe ao líder escolher uma solução, que poderá ser a dele ou não e essa será a solução do desafio para todos os jogadores. Por fim, cada jogador testa individualmente se a solução escolhida pelo líder é a solução do desafio.

Se a solução escolhida estiver errada, os jogadores podem optar por iniciar uma nova ronda, em que o líder passa a ser o primeiro jogador que aceitou o convite do *Jogador1*. Esta escolha de líder é rotativa, ou seja quando todos os jogadores forem líderes e o desafio ainda não for resolvido, a liderança volta para o primeiro jogador a ser nomeado líder, neste exemplo o *Jogador1* .

Quando a resposta escolhida por um líder for a solução do desafio, todos os jogadores envolvidos na resolução do desafio recebem uma recompensa e avançam para o próximo nível do jogo, saído do modo colaborativo.

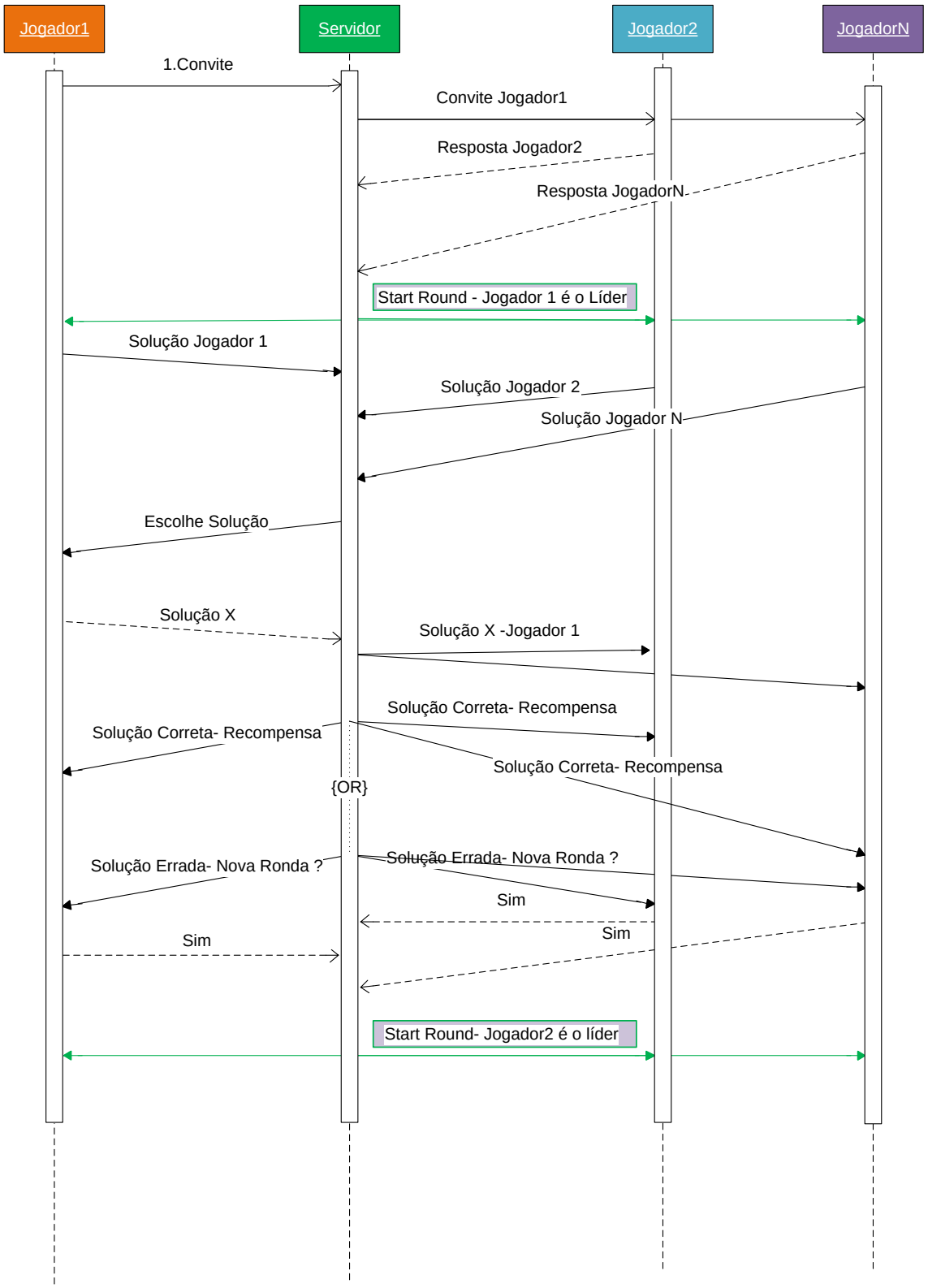

Figura 5.7: Diagrama Sequência processo colaboração no Boobo World.

5.4 Instruções e Ambiente de programação

Para resolver os desafios de programação e avançar no jogo, os jogadores terão de criar regras com as instruções definidas pela linguagem. Como já foi mencionado anteriormente estas instruções estão agrupadas em categorias, e cada uma delas têm o texto acompanhado de uma imagem, no mesmo contexto, facilitando assim a sua identificação e utilização.

Na Figura 5.8 estão duas categorias, os sensores e os filtros, utilizados na criação das condições.

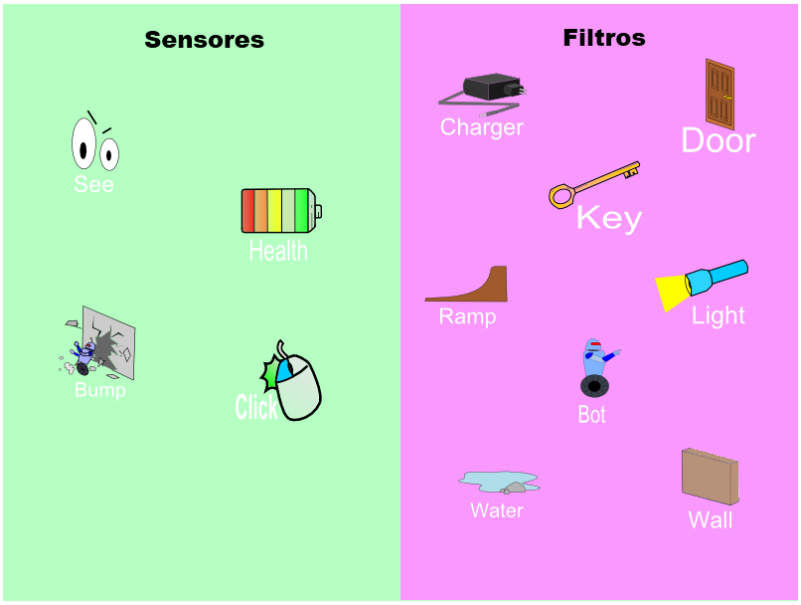

Figura 5.8: Instruções I - Sensores e Filtros.

As ações que o boobo pode efetuar durante o jogo podem ser vistos na Figura 5.9. Estas podem ser acompanhadas pelos seletores e modificadores, Figura 5.10.

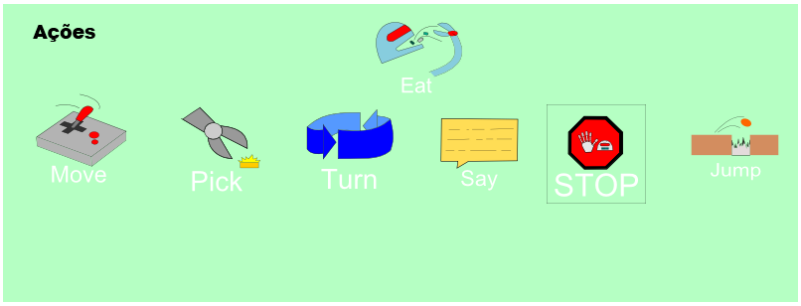

Figura 5.9: Instruções II - Ações.

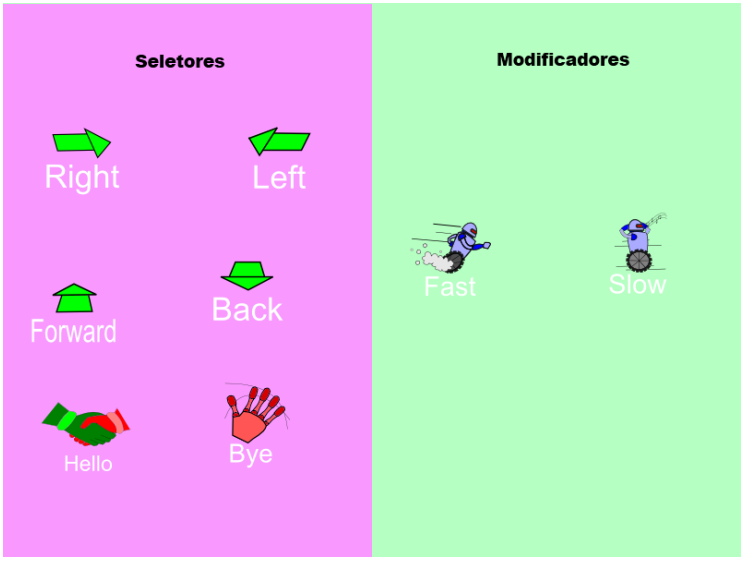

Figura 5.10: Instruções III - Seletores e Modificadores.

Todas estas instruções estão integradas num ambiente de programação simples e intuitivo, Figura 5.11. Este ambiente esta baseado no esboço apresentado no documento de game design (capítulo 4).

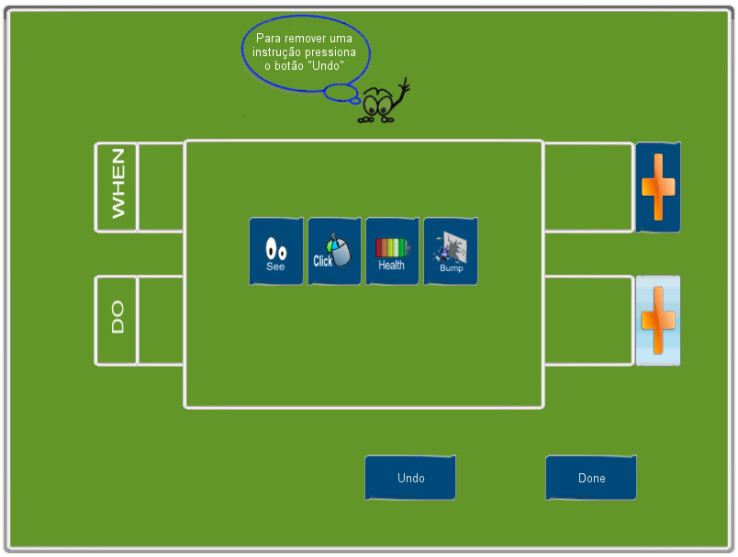

Figura 5.11: Ambiente programação no Boobo World.

Capítulo 6

Boobo World - Resultados e Avaliação

Os protótipos dos videojogos servem como prova de conceito ou uma forma de testar as ideias, adicionando, modificando ou removendo componentes dos mesmos [BS09]. E como tal, o protótipo do Boobo World desenvolvido nesta dissertação foi testado com o seu público alvo, as crianças.

Esta secção destina-se à apresentação do teste realizado, os resultados obtidos e uma avaliação dos mesmos.

6.1 Desenho do teste

Este teste consistiu em apresentar a crianças dois ambientes virtuais, cujo o principal foco é a aprendizagem dos conceitos básicos da programação. As crianças terão a possibilidade de resolver pequenos desafios de programação em cada um dos sistemas.

O objetivo deste teste é determinar a preferência das crianças, de um sistema em relação ao outro e identificar quais os pontos chaves da mesma. Estes dois ambientes são:

1. **Scratch:** É um ambiente de programação visual, onde os utilizadores são convidados a experimentarem os scripts e visualizar o resultado da sua execução, ver capítulo 3.

2. **Boobo World:** É o protótipo do jogo desenvolvido no âmbito desta dissertação. É um mundo virtual, onde cada criança com o seu avatar pode interagir com os objetos presentes e dessa forma aceder aos desafios de programação.

Os desafios que constam deste teste são (baseado nos três níveis do protótipo implementado):

- **Desafio 1** - Programar o robô para dizer algo, como por exemplo um "*olá*" ou um "*bye*".
- **Desafio 2** - Programar o robô para parar quando vê uma parede.
- **Desafio 3** - Programar o robô para voltar ao ponto de partida quando vê uma parede.

Estes desafios serão apresentadas as crianças em ambos os ambientes, primeiro no Scratch e depois no Boobo World.

Para além de observar o sucesso ou insucesso das crianças na resolução dos desafios nos dois ambientes, há um conjunto de métricas a serem utilizadas durante a realização do teste. Elas são:

- **Facilidade de interação** - Qual é apreciação das crianças em relação ao funcionamento do sistema, os botões, a disposição dos blocos de instrução, ...etc ?
- **Visual** - Como as crianças avaliam o aspeto visual do sistema, gráficos, cores, forma dos blocos de instrução, ...etc ?
- **Componente lúdica** - As crianças consideram os sistemas divertidos e apelativos ?
- **Programação** - Como as crianças reagem em relação à construção dos programas e rotinas de programação ?
- **Colaboração** - Qual é a opinião das crianças sobre o processo colaborativo no Boobo World ?

No anexo A está descrito o guião utilizado para realização deste teste bem como a folha utilizada para recolha dos resultados.

6.2 Realização do teste

Este teste foi realizado em duas sessões. A primeira envolvendo quatro crianças e a segunda seis. Cada sessão foi dividida em três partes, na primeira procedeu-se a apresentação das partes envolvidas e do teste. Depois procedeu-se a recolha de alguma informação das crianças, através de uma conversa informal e descontraída.

Na segunda parte, começou o teste, as crianças foram divididas em grupos(2 e 3 elementos) e procedeu-se a apresentação dos desafios e dos ambientes de programação. A quando da apresentação dos ambientes, as crianças tiveram tempo para se familiarizarem com os sistemas antes de passarem a resolução dos desafios propostos.

Por fim, na terceira parte, recolheu-se informações relativas a apreciação das crianças dos dois ambientes apresentados e dos desafios propostos. Cada criança deu a sua opinião sobre a preferência de um ambiente em relação ao outro, explicando as suas razões e propondo melhorias.

O resultado deste teste é apresentado na próxima secção.

6.3 Análise dos Resultados

Os resultados obtidos após a realização do teste, podem ser visualizados nas próximas tabelas e gráficos.

Na Figura 6.1 temos um gráfico com o número de participantes no teste e a sua distribuição por idade.

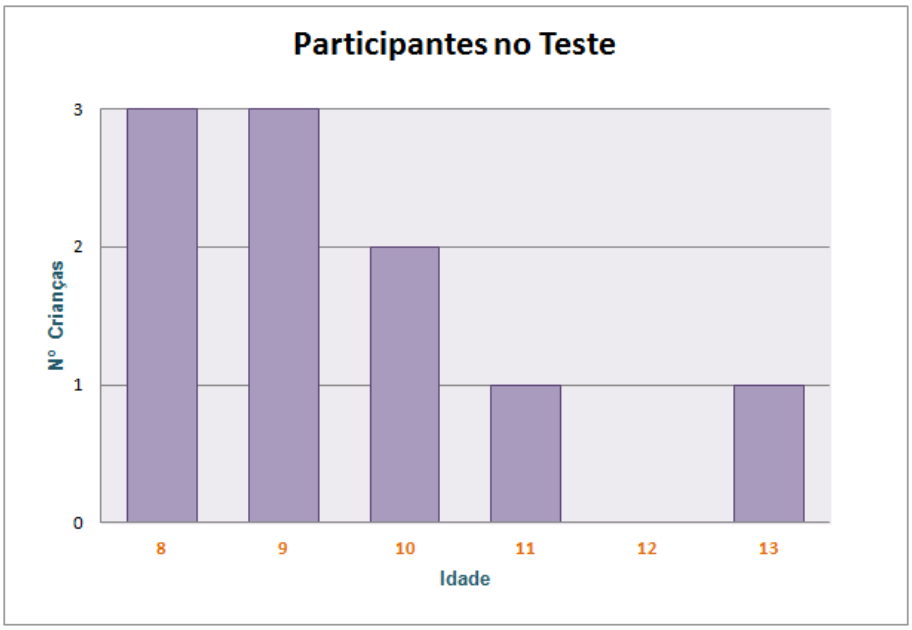

Figura 6.1: Participantes no teste.

Os desafios de programação propostos no teste, foram realizados em ambos o sistemas(Scratch e Boobo World) e o resultado pode ser visto na Figura 6.2, uma tabela, onde se pode constatar que as crianças tiveram dificuldades em realizar os desafios 2 e 3 no ambiente de programação Scratch, não conseguindo mesmo terminá-los com sucesso.

	Sistemas			
Desafios	Scratch		Boobo World	
	Sucesso	*Insucesso*	*Sucesso*	*Insucesso*
Desafio 1	+		+	
Desafio 2		-	+	
Desafio 3		-	+	

Figura 6.2: Performance das Crianças na resolução dos desafios.

Enquanto que no Boobo World, todos os desafios foram realizados com sucesso.

No final da realização do teste, as crianças puderam expressar as suas opiniões sobre os dois sistemas com o qual interagiram, e utilizando as métricas apresentadas anteriormente, o resultado encontra-se na Figura 6.3.

	Scratch	Boobo World
Facilidade de interação		x
Visual	x	
Componente lúdica		x
Programação		x
Apreciação geral		x

Figura 6.3: Scratch vs Boobo World - Preferências das crianças.

Todas as crianças envolvidas no teste mostraram-se mais à vontade a interagir com o Boobo World, e constatou-se também uma maior atração por este sistema do que o Scratch. As crianças consideraram o Boobo World mais "divertido" do que o Scratch e a programação como sendo mais intuitiva. A nível visual, arte, cores, formas, imagens entre outros o Scratch é o sistema completo e com uma maior número de possibilidades que o Boobo World, e a escolha das crianças foi clara, o Scratch.

Durante a realização do teste foi visível o nível de entusiasmo das crianças quando entraram no Boobo World pela primeira vez e puderam ver o avatar um dos outros a moverem-se no mesmo espaço. Esse entusiasmo aumentou ainda mais quando na resolução dos desafios, eles puderam colaborar entre si e não resolver os desafios individualmente como vinham a fazer.

O gráfico da Figura 6.4 mostra a opinião das crianças sobre o processo colaborativo na resolução dos desafios no Boobo World, e como pode-se observar a opinião e aceitação é positiva.

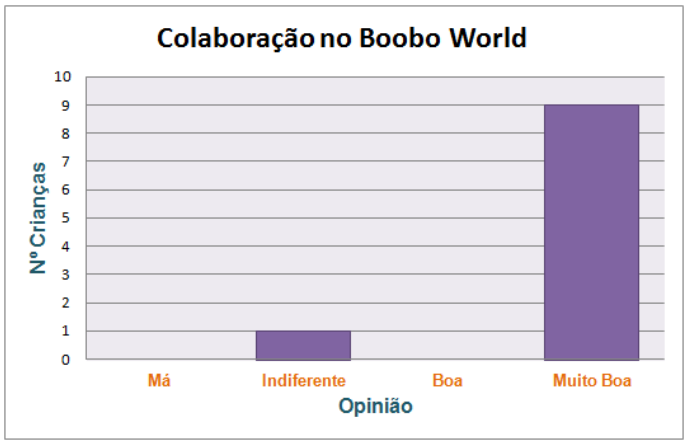

Figura 6.4: Opinião das Crianças em relação a Colaboração no Boobo World

Resumindo, este teste foi efetuado com sucesso e os resultados obtidos com este grupo de crianças é uma prova que o conceito de jogo foi bem conseguido e que caso o jogo estivesse completo este seria bem aceite.

6.4 Críticas e Sugestões

Um dos outros objetivos do teste realizado foi a recolha de sugestões e críticas por parte das crianças em relação ao protótipo do jogo desenvolvido.

As críticas foram:

- **Arte pouco evoluída** - As crianças sentiram falta de um ambiente de jogo mais elaborado a nível visual, com mais objetos com o qual pudessem interagir.
- **Ausência de um Chat** - O chat para poderem conversar com os amigos foi a *feature* no jogo que todos queriam ter.
- **Ausência de customização** - A possibilidade de customizar e personalizar o avatar e o robô, e de ter uma área pessoal no jogo foi outra componente no jogo que algumas crianças queriam ter.
- **Protótipo demasiado pequeno** - Todas as crianças fizeram uma observação sobre o tempo que demoraram a terminar o jogo. A expressão mais utilizada foi "Já acabou ?".

As sugestões foram baseadas críticas, mas houve uma em particular que pode ser uma mais valia para o jogo. Essa sugestão tem a ver coma possibilidade de cada jogador poder "*criar desafios*" e depois "*adicioná-los*" ao jogo.

Capítulo 7

Conclusões

Quando se fala em jogos de computador/videojogos as pessoas têm tendência a fazer uma associação direta com divertimento e lazer, e muito pouco trabalho. Contudo, esta associação não é tão linear, antes de um jogo chegar as mãos do jogador, há um conjunto de estados a serem percorridos. Estes estados (pensar, desenhar, criar, lançar o jogo,etc...) exigem estudo, trabalho e muita dedicação de quem produz videojogos. O criador do jogo, tirando algumas exceções, não é uma única pessoa, mas sim, várias equipas (*game designers*, programadores, artistas, etc...) a trabalhar, colaborar e cooperar entre si, de forma a terminar e lançar o jogo no mercado.

Por outro lado, quando se fala em ensino e aprendizagem não há tanto entusiasmo como no tema dos jogos. Este entusiasmo é ainda menor quando o que se quer ensinar é a programação, pois esta temática é considera, por muitos, como uma atividade mais técnica e só acessível a certos indivíduos. E se levarmos este conceito a crianças sem uma boa preparação, o resultado pode ser desolador.

Muitas pessoas poderão perguntar: Qual é o interesse em ensinar programação a crianças e não matemática, línguas ou ciências? A resposta a essa pergunta é: Quando as crianças aprendem os conceitos da programação, elas obtêm benefícios como, estruturar e organizar o pensamento, encontrar alternativas, etc..., que vão ajudá-los a aprender o resto com mais facilidade. Elas aprendem a saber fazer. E, do ponto de vista de um programador, a resposta a essa pergunta passa pela preocupação em garantir que alguém no futuro irá ser capaz de manter o código que este produz enquanto está vivo. E este alguém são as crianças, e quanto mais cedo começarem a aprender a programar melhor será. Durante a realização desta dissertação ficou claro que já houve várias tentativas para introdução dos conceitos de programação a crianças o mais cedo possível. No entanto, nem todas elas foram bem sucedidas.

O Scratch, Alice, Squeak Etoys e o Kodu, são sistemas ricos e que conseguiram tornar a programação mais acessível a crianças. São sistemas ricos em termos visuais e ferramentas que permitam as crianças construírem coisas que lhes interessem. Mas nenhuma dessas coisas é um jogo, e esta é a principal diferença para este projeto.

7.1 Satisfação dos Objetivos

Os objetivos propostos inicialmente foram cumpridos na sua totalidade, criou-se um documento de desenho do jogo, implementou-se um protótipo como prova de conceito para as ideias sugeridas e por fim fez-se uma validação do mesmo com o público alvo do jogo. Esta validação foi positiva e uma prova de que o conceito de jogo foi bem conseguido.

Com este projeto também é possível verificar que um jogo pode ser sério mas também muito divertido, conseguindo assim prender o jogador na resolução dos desafios de programação e aprendendo dessa forma os conceitos da programação.

7.2 Trabalho Futuro

O trabalho futuro passa pela finalização do jogo, nomeadamente uma afinação da linguagem de programação, construção de mais níveis e melhoramentos na arte do jogo. Estas alterações deverão ser acrescentado ao documento de *Game Design*.

Futuramente, será estudado a possibilidade do Boobo World, para além do ensino dos conceitos da programação, passar a incluir outras áreas de ensino como matemática, línguas e ciências.

Por fim, outro aspeto que poderá tornar o Boobo World num jogo mais apetecível, é a possibilidade de os próprios jogadores poderem gerar conteúdo para o jogo, isto é, criarem desafios e depois integrá-los no mundo virtual.

Anexo A

Materiais utilizados no teste

A.1 Guião do Teste

Contexto

Esta experiência consiste em apresentar as crianças dois ambientes virtuais, cujo o principal foco é a aprendizagem dos conceitos básicos da programação.

As crianças terão a possibilidade de resolver pequenos desafios de programação em cada um dos sistemas.

O objetivo desta experiência é determinar a preferência (single/multi user environment), das crianças, de um sistema em relação ao outro e identificar quais os pontos chaves da mesma.

Ambientes Virtual I - Scratch

É um ambiente de programação visual, onde os utilizadores são convidados a experimentarem os scripts e visualizar o resultado da sua execução. Para esta experiência foram pré-criados os 3 desafios de programação que as crianças terão de resolver criando os scripts da solução.

Ambientes Virtuail II - Boobo World

É o protótipo do jogo desenvolvido no âmbito desta dissertação. É um mundo virtual, onde cada criança com o seu avatar pode interagir com os objetos presentes e dessa forma aceder aos desafios de programação.

Desafios

- **Desafio 1** - Programar o Robot para dizer algo, como por exemplo um “olá” ou um “bye”.
- **Desafio 2** - Programar o Robot para parar quando vê uma parede.

- **Desafio 3** - Programar o Robot para voltar ao ponto de partide quando vê uma parede.

Introdução

- Fazer a minha apresentação : "Sou o Admilo, e hoje vamos fazer uma experiência altamente."
- Organizar as crianças nos respetivos lugares.

Parte I

1. "Que tipo de jogos gostas mais?"
2. "Gostas de jogar sozinho ou com os teus amigos(colaboração, cooperação ou competição)?"
3. "O que gostas mais nos jogos?"
4. "Achas que aprendes alguma coisa quando jogas, ou é só divertimento? O quê?"
5. "Conheces algum jogo/ferramenta em que és tu a decidir tudo o que acontece a tua personagem? Por exemplo tinhas um carro ou um boneco e tu metias o mexer como quisesses e quando quisesses?"
6. "Alguém conhece o Scratch?"
   (a) Sim: "Podes explicar aos teus colegas o que ê?"
   (b) Não: "O scratch é uma aplicação onde podes fazer os teus desenhos, e depois criar animações para eles, ou mesmo criar o teu próprio jogo."
7. Mostrar o ambiente, fazer uma pequena demonstração.
8. Pedir para resolverem o primeiro desafio
9. "Agora querem meter o robô a mexer? Desafio 2, Desafio 3".
10. "Gostaram do ambiente? Porquê?"
11. "Voçês agora estiveram aqui todos juntos a fazer isto, e se agora cada 1 estivesse no seu próprio pc,e mesmo assim conseguirem ver o que os outros estão a fazer? E se fosse um jogo?"

PARTE II

Apresentar o Boobo World.

1. “É um jogo em que tens um avatar(a tua personagem com o nome que lhe dás) e depois tens atrás um robot que não sabe fazer nada e depois tens que ensiná-lo a fazer as coisas.”
2. “Lembram se do primeiro desafio 1 no scratch? Neste jogo tem o mesmo, vamos fazer.”
3. “E agora vamos fazer o segundo em conjunto, mas cada 1 no seu pc. Um jogador convida os outros todos, depois todos respondem, e este jogador escolhe a resposta que ache que esta certa. Se ele errar não tem mal, voltam a jogar e agora passa outro jogador a escolher a resposta. Querem experimentar?”
4. “Gostaram do jogo? Porquê?”
5. “Acabou, obrigado!”

A.2 Folha de Recolha de Resultados

(Parte I)

Nº de Participantes ___

Idades	6	7	8	9	10	11	12	13

Iniciar a sessão falando sobre os videojogos e apurar as preferências dos participantes por jogos:

SinglePlayer ___
Multiplayer ___

Recolher a preferência dos tipos de jogos de cada participante:

Ação ___
Estratégia ___
Simulação ___
Role-playing ___
Outro ___

Quais os elementos dos jogos que cada participante gosta mais:

Arte ___
Avatar ___
Mecânica___
Recompensas__

Fazer uma primeira abordagem sobre os jogos Sérios, perguntando a cada participante se acham que os videojogos lhes ensinam alguma coisa:

Sim ___
Não___

Figura A.1: Folha de recolha de resultados 1/3

(Aos que responderem que sim)
O quê?

Falar sobre os jogos Sérios para Educação(Learning), e averiguar os títulos que cada participanete mais gosta:

Matemática ___
Línguas __
história__
Ciências __
Arte e Cultura ___

(Parte ||)

Determinar se os participantes tem alguma noção do conceito "Programação"!

Sim __
Não __

Apresentar aos participantes o ambiente de programação Scratch e deixá-los interagir com o sistema, e analisar os seguintes pontos:

	1	2	3	4	5
Facilidade de interação					
Visual					
Componente lúdica					
Programação					
Compreensão geral					

Os participantes perceberam os objetivos do Scratch?

Sim__ Não ___

Qual é opinião dos participantes, se o Scratch tivesse a possibilidade de multiplayer?

Figura A.2: Folha de recolha de resultados 2/3

Má __ Boa___ Indiferente__ Muito Boa ___ Excelente ____

Parte(|||)

Apresentar o Boobo World, e utilizar as mesmas métricas utilizadas na apresentação do Scratch:

	1	2	3	4	5
Facilidade de interação					
Visual					
Componente lúdica					
Programação					
Compreensão geral					

Os participantes perceberam os objetivos do Boobo World?
Sim__ Não ___

O que gostaram no Boobo World?
Boobo___
Avatar ___
Ambiente Programação ___
Desafios ____
Colaboração ___

Aspetos a melhorar no Boobo World?

Figura A.3: Folha de recolha de resultados 3/3

Materiais utilizados no teste

Referências

[Age11] Inc. AgentSheets. What is agentsheets?, 2011. Disponível em http://www.agentsheets.com/products/index.html, acedido a última vez em 05 de Julho de 2011.

[Ali10] Alice.Org. What is alice?, 2010. Disponível em http://www.alice.org/index.php?page=what_is_alice/what_is_alice, acedido a última vez em 03 de Julho de 2011.

[App06] T. H. Apperley. Genre and game studies: Toward a critical approach to video game genres. *Simulation & Gaming*, 37(1):6–23, March 2006.

[Bai89] J.H. Bair. Supporting cooperative work with computers: addressing meeting mania. In *COMPCON Spring '89. Thirty-Fourth IEEE Computer Society International Conference: Intellectual Leverage, Digest of Papers.*, pages 208 –217, feb-3 mar 1989.

[BD05] Curtis J Bonk e Vanessa P Dennen. Technical Report 2005-1 Massive Multiplayer Online Gaming : A Research Framework for Military Training and Education OFFICE OF THE UNDER SECRETARY OF DEFENSE READINESS AND TRAINING , OFFICE OF THE UNDER SECRETARY. *Director*, (March), 2005.

[Bru97] Amy Susan Bruckman. *MOOSE Crossing: Construction, Community, and Learning in a Networked Virtual World for Kids*. PhD thesis, MIT, 1997.

[BS09] B. Brathwaite e I. Schreiber. *Challenges for game designers*. Course Technology, 2009.

[CB98] A. Cockburn e A. Bryant. Cleogo: Collaborative and multi-metaphor programming for kids. In *Proceedings of the Third Asian Pacific Computer and Human Interaction*, pages 189–, Washington, DC, USA, 1998. IEEE Computer Society.

[CD00] Stephen Cooper e Wanda Dann. Alice: a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in*, 5(May 2000):108–117, 2000.

[Cor06] Microsoft Corporation. Microsoft flight simulator x, 2006. Disponível em http://www.microsoft.com/games/flightsimulatorx/, acedido a última vez em 02 de Julho de 2011.

[Der07] Anne Derryberry. Serious games: online games for learning. *Retrieved from http://www. adobe. com/resources/*, 2007.

[Fou00] Logo Foundation. What is logo?, 2000. Disponível em http://el.media.mit.edu/logo-foundation/logo/index.html, acedido a última vez em 05 de Julho de 2011.

[Fou11] Logo Foundation. Logo products, 2011. Disponível em http://el.media.mit.edu/logo-foundation/products/software.html, acedido a última vez em 05 de Julho de 2011.

[Gal09] Matthew Gallant. Mechanics, dynamics and aesthetics, 2009. Disponível em http://gangles.ca/2009/08/21/mda/, acedido a última vez em 29 de Junho de 2011.

[HK04] Poul Henriksen e Michael Kölling. greenfoot: combining object visualisation with interaction. In *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, OOPSLA '04, pages 73–82, New York, NY, USA, 2004. ACM.

[HLZ04] Robin Hunicke, M. LeBlanc e Robert Zubek. MDA: A formal approach to game design and game research. In *Proceedings of the AAAI-04 Workshop on Challenges in Game AI*, pages 1–5, 2004.

[Inc11] Club Penguin Disney Online Studios Canada Inc. Parent's guide, 2011. Disponível em http://www.clubpenguin.com/parents/club_penguin_guide.htm, acedido a última vez em 06 de Julho de 2011.

[Int10] Spongelab Interative. Genomics digital lab history of biology, 2010. Disponível em http://www.genomicsdigitallab.com/gdl/default.cfm?thePage=CA/about#gamebased, acedido a última vez em 02 de Julho de 2011.

[IRW+10] Andri Ioannidou, Alexander Repenning, David Webb, Diane Keyser, Lisa Luhn e Christof Daetwyler. Mr. vetro: A collective simulation for teaching health science. *International Journal of Computer-Supported Collaborative Learning*, 5:141–166, 2010. 10.1007/s11412-010-9082-8.

[Jen09] Henry Jenkins. *Confronting the challenges of participatory culture: Media education for the 21st century*. The MIT Press, 2009.

[KÖ8] Michael Kölling. Greenfoot: a highly graphical ide for learning object-oriented programming. *SIGCSE Bull.*, 40:327–327, June 2008.

[Kah00] Ken Kahn. Toontalk - um jogo para criar jogos de computador, 2000. Disponível em http://www.toontalk.com/pt/toontalk.htm, acedido a última vez em 04 de Julho de 2011.

[Kid12] Driving Kids. Driving kids - free online education games for kids, 2012. Disponível em http://www.drivingKids.com, acedido a última vez em 11 de Janeiro de 2012.

[KK05] Alan Kay e Alan Kay. Squeak Etoys , Children & Learning Squeak Etoys , Children & Learning. *Simulation*, (818), 2005.

[KP05] Caitlin Kelleher e Randy Pausch. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Comput. Surv.*, 37:83–137, June 2005.

[Ltd08] Mind Candy Ltd. Moshi monsters parents, 2008. Disponível em http://www.moshimonsters.com/parents, acedido a última vez em 06 de Julho de 2011.

[LW08] D.J. Lubliner e G. Widmeyer. N2 heads are better than one: Collaborative learning, utilizing an integrated; knowledge repository, facilitated through a massively multiplayer online gaming (mmog) paradigm. In *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual*, page 350, jan. 2008.

[MRR+10] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman e Evelyn Eastmond. The scratch programming language and environment. *Trans. Comput. Educ.*, 10:16:1–16:15, November 2010.

[MSS04] Alice Mitchell e Carol Savill-Smith. The use of computer and video games for learning: A review of the literature. page 93, 2004.

[myG11] myGames. Kojima: "termo videojogo deixará de existir", 2011. Disponível em http://ez.mygames.pt/ps3/accao/metal-gear-solid-hd-collection/noticia/kojima-termo-videojogo-deixara-de-existir-14935/, acedido a última vez em 02 de Julho de 2011.

[NK08] Brian Nelson e Diane Ketelhut. Exploring embedded guidance and self-efficacy in educational multi-user virtual environments. *International Journal of Computer-Supported Collaborative Learning*, 3:413–427, 2008. 10.1007/s11412-008-9049-1.

[Ord11] Juan P. Ordóñez. Videojogos 2011 - workshop on game design, 2011.

[Pan97] T. Panitz. Collaborative versus cooperative learning: A comparison of the two concepts which will help us understand the underlying nature of interactive learning. *Cooperative Learning and College Teaching*, 8(2):5–7, 1997.

[PI11] S.A. Priberam Informática. Dicionário priberam da língua portuguesa, 2011. Disponível em http://www.priberam.pt/dlpo/default.aspx?pal=colaborar, acedido a última vez em 05 de Julho de 2011.

[Pre01] M. Prensky. The digital game-based learning revolution. *Digital game-based learning*, pages 1–19, 2001.

[Pre02] Marc Prensky. The motivation of gameplay: The real twenty-first century learning revolution. *On the Horizon*, 10(1):5–11, 2002.

[Pre03] Marc Prensky. Digital game-based learning. *Comput. Entertain.*, 1:21–21, October 2003.

[QN08] Clark Quinn e Lisa Neal. Serious games for serious topics. *eLearn*, 2008:5:1–5:1, March 2008.

[Res11] Microsoft Research. Kodu, 2011. Disponível em http://research.microsoft.com/en-us/projects/kodu/, acedido a última vez em 12 de Outubro de 2011.

[Ric11] Rich. The reality of developing web games with flash, html5 and unity, November 2011. Disponível em http://alturl.com/2ykh3, acedido a última vez em 03 de Janeiro de 2012.

[RMMH+09] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman e Yasmin Kafai. Scratch: programming for all. *Commun. ACM*, 52:60–67, November 2009.

[Sci10] Eytam Computer Science. Mama documentation, 2010. Disponível em http://www.eytam.com/mama/doc, acedido a última vez em 04 de Julho de 2011.

[SHO03] Jouni Smed, Harri Hakonen e Others. Towards a definition of a computer game. *Science*, (553), 2003.

[SS08] B. Sawyer e P. Smith. Serious games taxonomy. In *Slides from the Serious Games Summit at the Game Developers Conference*, pages 1–54, 2008.

[ST05] SwissEduc-Team. Kara - programming with state machines, 2005. Disponível em http://www.swisseduc.ch/compscience/karatojava/kara/, acedido a última vez em 04 de Julho de 2011.

[Ste04] Constance A. Steinkuehler. Learning in massively multiplayer online games. In *Proceedings of the 6th international conference on Learning sciences*, ICLS '04, pages 521–528. International Society of the Learning Sciences, 2004.

[Sto10] Kathryn T. Stolee. Kodu language and grammar specification, microsoft research whitepaper, September 2010.

[Tee08] R. Teed. Serc. carleton college starting point. what makes a good game?, 2008. Disponível em http://serc.carleton.edu/introgeo/games/goodgame.html, acedido a última vez em 13 de Julho de 2011.

[Tra96] Michael David Travers. *Programming with Agents: New metaphors for thinking about computation*. PhD thesis, MIT, 1996.

[uni11] unity3D. High level networking concepts, 2011. Disponível em http://unity3d.com/support/documentation/Components/net-HighLevelOverview.html, acedido a última vez em 20 de Dezembro de 2011.

[uni12] unity3D. Publishing, 2012. Disponível em http://unity3d.com/unity/publishing/consoles.html, acedido a última vez em 03 de janeiro de 2012.

[Wik11a] Wikipédia. Flight simulator, 2011. Disponível em http://en.wikipedia.org/wiki/Flight_simulator, acedido a última vez em 02 de Julho de 2011.

[Wik11b] Wikipédia. Genomics digital lab, 2011. Disponível em http://en.wikipedia.org/wiki/Genomics_Digital_Lab, acedido a última vez em 02 de Julho de 2011.

[Wik11c] Wikipédia. Multiplayer game, 2011. Disponível em http://en.wikipedia.org/wiki/Multiplayer_game, acedido a última vez em 14 de Julho de 2011.

[Wik11d] Wikipedia. Mda framework, 2011. Disponível em http://en.wikipedia.org/wiki/MDA_framework, acedido a última vez em 29 de Junho de 2011.

[Wik11e] Wikipedia. Personal computer game, 2011. Disponível em http://en.wikipedia.org/wiki/Personal_computer_game, acedido a última vez em 29 de Junho de 2011.

[Wik11f] Wikipedia. Video game genres, 2011. Disponível em http://en.wikipedia.org/wiki/Video_game_genres#Other_notable_genres, acedido a última vez em 01 de Julho de 2011.

[Wil05] Joseph N. Wilson. About karel's world, 2005. Disponível em http://www.cise.ufl.edu/~jnw/Karel/about-karel%27s-world.html, acedido a última vez em 04 de Julho de 2011.