UNIVERSIDADE LUSÓFONA
de Humanidades e Tecnologias
Humani nihil alienum

Escola de Comunicação, Artes e Tecnologias da Informação

Relatório de Trabalho Final de Curso

Tema:

The Squared One

Jogo Windows 2D de Plataformas

Aluno: Ruben Emanuel Machado da Silva

Orientador: Professor Sérgio Guerreiro

Indíce

Resumo do Trabalho

Este trabalho consiste num jogo para o sistema operativo Windows de plataformas em 2D e tem por objectivo conduzir a personagem principal do jogo do início ao fim de cada nível utilizando as setas direccionais e a barra de espaços do teclado. Desta forma, o utilizador terá de adoptar a melhor estratégia para acumular o maior número de pontos possíveis e derrotar os vários inimigos existentes nos diversos níveis, sendo que a dificuldade aumenta à medida que o utilizador vai progredindo no jogo. Para garantir que o utilizador cumpre o objectivo a que se propõe, pode aumentar a sua pontuação colecionando os vários tipos de gemas ou destruíndo os inúmeros adversários saltando múltiplas vezes em cima dos mesmos.

Abstract

This project is a 2D platform game for the Windows operating system and the main goal is to lead the main character of the game from the start to the finish of each level by using the arrow keys and the spacebar on the keyboard. Thus, the user will have to adopt the best strategy to accumulate as many points as possible and defeat the various enemies within the different levels, and the difficulty increases as the user progresses in the game. To ensure that the user fulfills the purpose for which it is intended, it is possible to increase your score by collecting various types of gems or destroying the numerous opponents jumping multiple times on them.

1. Introdução

O computador é uma máquina incrível, capaz de inúmeras formas de tratamento automático de informação ou processamento de dados. É uma ferramenta utilizada por milhões de pessoas para armazenamento de dados, cálculo em grande escala, desenho industrial, tratamento de imagens gráficas, realidade virtual, cultura, entretenimento, entre outras.

Dentro da realidade virtual/entretenimento está a área de desenvolvimento de videojogos. Os videojogos são um assunto bastante complexo, embora muitas pessoas os vejam como meros meios utilizados para a diversão.

Básicamente, estes baseiam-se numa ferramenta de "input", tais como: um comando, um rato e teclado, um joystick, entre outros e outra ferramenta desta vez de "output", tais como: um monitor de computador, uma televisão, entre outros, sendo que o utilizador pode interagir de uma forma simples e divertida com o ambiente que lhe é proporcionado.

Os videojogos também utilizam elementos adicional, tais como: sons ou vibrações; de modo a tornarem-se mais atractivos e engraçados para o utilizador, sendo que conseguem desta forma simular de forma quase perfeita um ambiente totalmente diferente ao que o utilizador se encontra.

Por sua vez, os videojogos já são desenvolvidos há decadas e, portanto, a necessidade de satisfazer o cada vez mais exigente utilizador é sempre crescente e, deste modo, quando são criados, os melhores videojogos são os que apresentam o melhor balanço entre conteúdo e história.

Dentro dos videojogos, existem imensas categorias: jogos de acção, jogos de aventura, jogos de estratégia, entre outros. Uma categoria icónica no mundo dos videojogos é os jogos de plataformas. Jogos como Donkey Kong ou Super Mario Bros ou até mesmo Sonic são os jogos mais conhecidos dentro desta categoria, sendo mesmo considerados jogos clássicos por grande parte dos entusiastas dos videojogos.

Deste modo, foi baseado nestes três grandes jogos referidos anteriormente que o jogo apresentado neste relatório foi sendo desenvolvido. Cada um destes três jogos clássicos tem o ou os seus pontos fortes e, durante todo o processo de desenvolvimento deste jogo, existiu sempre a preocupação de reunir as melhores ideias de cada jogo e juntá-las todas em apenas um jogo.

Partindo de tudo isto, chega-se à questão: o que é necessário para se desenvolver um videojogo de plataformas?

A partir daqui, ir-se-á demonstrar neste relatório a resposta a esta pergunta.

2. Enquadramento Teórico

A motivação para o tema deste trabalho surgiu aquando da aprendizagem da cadeira de Computação Gráfica, mais específicamente a componente prática da cadeira, que se focava no desenvolvimento de um jogo para a plataforma Windows e, desta forma, também este trabalho se foca no desenvolvimento de um jogo para a plataforma Windows.

No começo, a ferramenta escolhida para o desenvolvimento do jogo foi Microsoft Visual Studio 2010 e Microsoft XNA Game Studio 4.0 utilizando a linguagem C#, mas devido ao facto de existirem ferramentas mais divertidas e intuitivas para o desenvolvimento de jogos, foi tomada a decisão de utilizar a ferramenta Game Maker Studio.

Contudo, embora a ferramenta facilite em alguns aspectos o desenvolvimento de um jogo, exige inúmeros conceitos leccionados nas várias cadeiras do curso, nomeadamente Fundamentos de Programação e Liguagens de Programação I e II, onde foram ensinados os conceitos básicos da linguagem C e JAVA, cruciais para todo o processo de desenvolvimento do jogo, visto que existem bastantes semelhanças na sintaxe da linguagem C, JAVA e GML (Game Maker Language).

De qualquer forma, como qualquer outra linguagem de programação existente, tem detalhes que a distinguem de todas as outras linguagens e, desta forma, também foi necessário a total compreensão desta linguagem, o GML.

3. Método

Etapas:

3.1 Escolha da Ferramenta

No início deste projecto, foi efectuado uma pesquisa na Internet sobre possíveis ferramentas a utilizar com o objectivo de desenvolver um jogo para a plataforma Windows, tendo obtido como principal resultado a ferramenta Game Maker Studio. Após uma discussão com o Orientador, foi proposto por este o uso de outra ferramenta, Microsoft Visual Studio 2010 e Microsoft XNA Game Studio 4.0, tendo sido esta última ferramenta a seleccionada para as primeiras abordagens ao desenvolvimento do jogo.

Desta forma, os primeiros protótipos revelaram-se de difícil desenvolvimento e demasiado dispendiosos, em termos de tempo, sendo que o resultado obtido foram jogos básicos e com poucas funcionalidades. Mesmo assim, o último protótipo realizado utilizando a linguagem C#, incluía algumas funcionalidades consideradas de cruciais no processo de desenvolvimento do jogo, tais como: desenho do mapa e movimento do personagem principal.

No entando, o desenvolvimento do jogo utilizando a linguagem C# estava a ser feito principalmente com recurso a uma lista de vídeos disponibilizados na página (http://www.youtube.com/playlist?list=PLE500D63CA505443B), mas, mesmo assim, à medida que a complexidade do jogo aumentava, inúmeros obstáculos surgiam forçando o desenvolvimento do jogo a parar.

Contudo, após nova discussão com o Orientador, foi decidido que a opção mais racional a tomar seria a substituição da ferramenta Microsoft Visual Studio 2010 e Microsoft XNA Game Studio 4.0 pela ferramenta Game Maker Studio.

Na página (http://www.yoyogames.com/gamemaker/studio) está descrita a especialização da ferramenta Game Maker Studio para o desenvolvimento de jogos maioritariamente 2D para diversas plataformas sendo a plataforma Windows a mais importante para o trabalho, pelo que foi a ferramenta escolhida.

Esta ferramenta possibilita, como já foi dito, criar jogos para múltiplas plataformas, tais como: Windows, Mac OS X, HTML5, iOS, Android, entre outras mediante a compra de cada uma destas extensões. Para o desenvolvimento do meu jogo, a extensão necessária é Windows com o objectivo de testar e, eventualmente, gerar o jogo.

A linguagem utilizada nesta ferramenta é exclusiva da mesma denominada de Game Maker Language ou GML e, tal como a linguagem C e JAVA, é orientada a objectos.

3.2 A Ferramenta Game Maker Studio

A ferramenta Game Maker Studio foi desenvolvida com recurso à linguagem de programação Delphi com o objectivo de desenvolver jogos para diversas plataformas.

Para principiantes que estão a começar a investigar o mundo por de trás do desenvolvimento de jogos, esta ferramenta oferece uma interface bastante amiga do utilizador, sendo que este consegue construir pequenos protótipos de jogos básicos utilizando apenas o método de "arrastar e largar".

Contudo, para utilizadores mais avançados, esta ferramenta também prova ser muito útil, visto que todas as funcionalidades disponíveis na caixa de ferramentas do próprio Game Maker Studio são também acessíveis através de instruções de código inseridas em scripts, funcionalidades essas que, utilizando instruções de código , é possível ter um maior controlo e muito mais liberdade sobre o que realmente acontece durante determinada situação.

Comparando a ferramenta do Game Maker Studio directamente com a ferramenta Microsoft Visual Studio 2010 e Microsoft XNA Game Studio 4.0, foi concluído que: em primeiro lugar, a construção dos protótipos iniciais foi completada de forma mais fácil e intuitíva utilizando o Game Maker Studio, visto que, para efeitos de teste, a utilização da caixa de ferramentas disponível nesta ferramenta poupa bastante tempo em escrita de código; em segundo lugar, a criação dos mapas para o jogo, a colocação dos vários objectos (jogador, inimigos, entre outros), o acrescento de elementos adicionais, tais como: fundos de ecrã ou sons e a edição tanto de imagens como de sons é tudo possível dentro da mesma ferramenta, pelo que é seguro concluir que o Game Maker Studio é uma ferramenta extremamente eficiente e produtiva; e em terceiro lugar, devido ao facto de a comunidade utilizadora desta ferramenta ser tão prestável, é possível obter ajuda através do website da própria ferramenta, caso que, pelo que foi comprovado, não acontece quando foi utilizada a ferramenta Microsoft Visual Studio 2010 e Microsoft XNA Game Studio 4.0.

No entando, nem tudo nesta ferramenta é excelente. Também foram encontradas algumas desvantagens no uso desta ferramenta: primeiro, devido ao facto de a ferramenta gerar o jogo para as diversas plataformas automáticamente, para algumas plataformas o ficheiro de instalação e o ficheiro do próprio jogo são de certa forma um pouco pesados em termos de espaço ocupado, sendo neste ponto esta ferramenta ultrapassada por outras ferramentas concorrentes; e por último, tendo em conta que esta ferramenta utiliza uma linguagem exclusiva, existe a necessidade de aprender todos os seus detalhes importantes para o desenvolvimento do jogo obrigando assim a visitas frequentes aos manuais da própria linguagem GML e, tal como referido anteriormente, à grande comunidade de desenvolvedores activa no website da ferramenta.

Relativamente à linguagem GML, esta permite que utilizadores mais avançados possam controlar e intensificar certos aspectos do jogo através de programação convencional e, desta forma, criar funcionalidades muito mais avançadas e complexas que ou são muito confusas e pouco eficientes de implementar ou são simplesmente inacessíveis com recurso à caixa de ferramentas.

Tal como já foi referido, a sintaxe da linguagem GML tem bastantes semelhanças com as linguagens de programação: C, C++ e JAVA, sendo que todas estas possibilitam a programação orientada a objectos. Possui também determinadas funções e variáveis já implementadas na própria linguagem GML e permite executar ficheiros no formato DLL (Biblioteca de Vínculo Dinâmico), sendo assim possível executar porções de código noutras linguagens de programação.

3.3 Construir e Desenvolver o Jogo

Ecrãs de Jogo:

A estrutura do jogo foi idealizada com três tipos diferentes de ecrãs:

- Ecrã de Menu Principal;
- Ecrã de Fim;
- Ecrã de Nível.

Desta forma, para ser possível criar estes ecrãs, foram necessárias inúmeras imagens com o objectivo de criar botões, a personagem principal, inimigos, entre outros, imagens essas que com o auxílio do rato e teclado e algum código fonte por detrás ganham vida e desencadeiam diferentes acções.

Ecrã de Menu Principal:

Para ser possível criar este ecrã, foi necessário um simples conjunto de passos:

- Primeiro, foi necessário reunir algumas imagens, nomeadamente: uma imagem para o fundo de ecrã e uma imagem de um botão "Play". A imagem para o fundo de ecrã idealmente teria de estar de alguma forma relacionada com o contexto do jogo e a imagem do botão "Play" idealmente teria de ter uma simples animação para embelezar o ecrã. As duas imagens foram encontradas na Internet através de algumas pesquisas no conhecido motor de buscas Google.
- De seguida, foi criada a "Room" (nome atribuído a cada ecrã no jogo) e recorrendo às imagens encontradas foi adicionado um fundo de ecrã.

- Quanto à imagem do botão “Play”, esta foi adicionada como “Sprite” de um objecto com o objectivo de que este cumprisse a tarefa de que quanto um utilizador clique em cima desta imagem, o actual ecrã de jogo deixa de ser o ecrã de menu principal e passa a ser o ecrã, neste caso, do primeiro nível e dá assim início ao jogo.
- Posto isto, faltava apenas adicionar um título, título esse que foi rápidamente adicionado recorrendo à ferramenta Paint.NET.

O desenvolvimento deste simples ecrã não levantou grandes problemas, vale apenas fazer referência a um pequeno problema que surgiu derivado do facto de ter sido decidido colocar uma simples animação no botão “Play” para embelezar o ecrã. A solução, também ela simples, foi encontrada depois de alguma pesquisa na Internet. Foi dividido o único evento de clique do rato em vários eventos de forma a conseguir alterar a “Sprite” do botão em questão consoante o clique e/ou posição do rato.

Ecrã de Fim:

À semelhança do ecrã de menu principal, para ser possível criar este ecrã, foi necessário um simples conjunto de passos:

- Foi necessário reunir outras imagens, com o objectivo destas desempenharem funções muito idênticas às anteriores. Uma imagem para o fundo de ecrã e outra imagem de um botão.
- Tudo idêntico ao ecrã de menu principal, com a excepção de que a acção desencadeada pelo botão neste ecrã, faz com que o actual ecrã de jogo deixe de ser o ecrã de fim e volte novamente ao ecrã de menu principal.

Ao contrário do que se passou no ecrã de menu principal, o desenvolvimento deste ecrã levantou alguns problemas. O problema mais relevante foi: como foi decidido que a forma mais original de terminar o jogo seria aparecer o ecrã de fim enquanto é permitido ao utilizador continuar a movimentar a personagem principal, apenas após o utilizador clicar com o rato no botão é que o objecto Personagem Principal deve ser destruído. A solução para este problema passou por adicionar um novo evento ao botão, evento esse que, quando o utlizador clicar com o rato, destrói o objecto Personagem Principal juntamente com todos os outros objectos associados a ele.

Ecrã de Nível:

Este foi, sem qualquer dúvida, o ecrã que demorou mais tempo a ser criado por várias razões:

- Em primeiro lugar, sendo o jogo do tipo de plataformas, foi crucial a criação de cinco níveis diferentes encadeados, ou seja, quando um utilizador completa o primeiro nível, é imediatamente colocado no início do segundo nível. Desta forma, foram necessárias inúmeras imagens para cada um dos objectos que compõem o jogo e, sendo assim, foi realizada uma pesquisa na Internet de forma a conseguir reunir todas as imagens necessárias, nomeadamente imagens para: a personagem principal, gemas, plataformas, inimigos, entre outras.
- De seguida, foram criados os diversos objectos referidos anteriormente com a sua respectiva "Sprite". Relativamente às funcionalidades dos diferentes objectos, irá ser tudo explicado mais à frente neste relatório.
- Em terceiro lugar, foi realizada uma nova pesquisa na Internet, esta com o objectivo de encontrar alguns fundos de ecrã indicados para um jogo desta natureza.
- Assim, a última etapa nesta fase de desenvolvimento do jogo, foi criar as diferentes "Rooms", uma para cada nível, e popular estas com os todos os objectos criados.

O desenvolvimento deste ecrã, tal como já foi referido anteriormente neste relatório, foi o que demorou mais tempo e, desta forma, levantou diversos problemas, problemas esses que vão ser explicados mais à frente neste relatório indicando também qual foi a respectiva solução encontrada.

Nível 1:

Nível 2:

Nível 3:

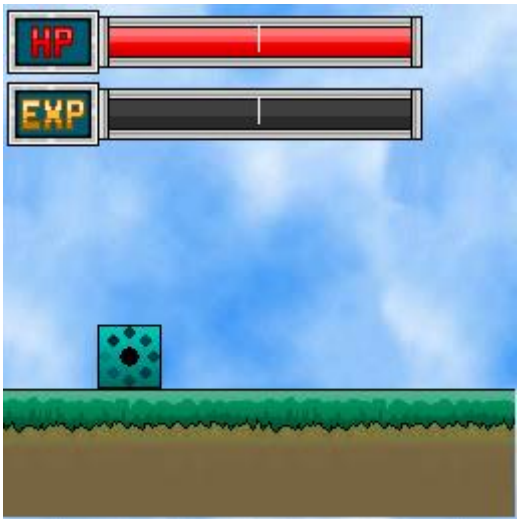
Nível 4:

Nível 5:

3.4 Funcionalidades Avançadas

Neste segmento do relatório vai ser explicado em detalhe cada um dos inúmeros objectos criados juntamente com todas as funcionalidades implementadas nestes.

Personagem Principal:



Para começar, foram criadas e inicializadas dentro do objecto da personagem principal (obj_player) várias variáveis a ser utilizadas ao longo de todo o processo de criação deste objecto. O passo seguinte foi recorrendo a essas variáveis e procedendo a uma série de verificações, foram então implementadas as diversas funcionalidades que compõem a personagem principal. De seguida, as funcionalidades implementadas mais relevantes:

- Uma verificação para controlar se a personagem principal está ou não em contacto com algum tipo de superfície e, caso não esteja, activa o sistema de gravidade do jogo. O sistema de gravidade do jogo teve de ser cuidadosamente codificado e testado, visto que é responsável por forçar a personagem principal a voltar a entrar em contacto com o solo e caso o valor definido para a variável referente à gravidade existente no jogo não fosse o indicado, afectaria directamente a capacidade de salto da personagem principal;
- Uma verificação para controlar a velocidade máxima que a personagem principal pode atingir, de forma a evitar pequenas falhas na verificação de colisões. Esta verificação foi obrigatóriamente criada devido a um problema encontrado, problema esse que, à medida que o utilizador continuasse a pressionar supondo na seta direccional direita, o jogo vai aumentando gradualmente o valor da variável referente à deslocação da personagem principal na horizontal e, atingindo este um valor suficientemente alto, a personagem principal, em vez de se deslocar suavemente ao longo das diversas plataformas, seria possível observar inúmeros saltos na posição da mesma;
- Uma série de verificações para controlar quais as teclas associadas ao movimento da personagem principal estão a ser premidas e desencadear, por sua vez, o movimento correspondente (esquerda, direita ou salto);
- Uma verificação para controlar se a personagem principal se encontra em contacto com alguma plataforma, com água ou se está no ar. Esta verificação é bastante importante, pelo facto de controlar quando é que a personagem principal pode ou não saltar. Caso esteja em contacto com alguma plataforma, é permitido que esta salte. Caso esteja na água, é permitido que esta salte múltiplas vezes e as variáveis associadas ao movimento desta são reduzidas sensívelmente para metade com o objectivo de simular a personagem principal a nadar. Caso esteja no ar, não é permitido que esta salte até q entre de novo em contacto com qualquer plataforma ou água,

sendo que a tarefa de forçar que a personagem principal volte a estar em contacto com alguma superfície é levada a cabo pelo sistema de gravidade;

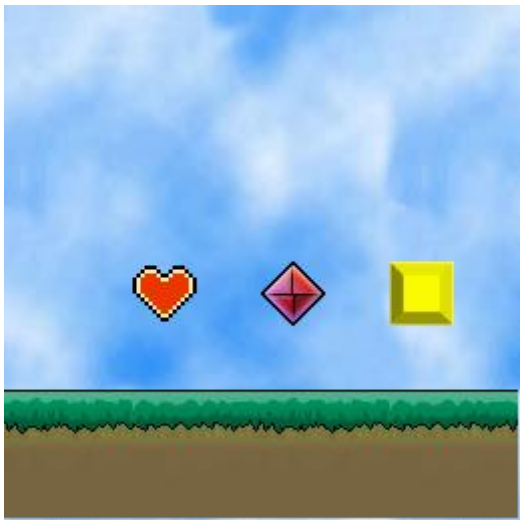
- Uma verificação crucial efectuada sempre antes que exista qualquer tipo de movimento da personagem principal para controlar se na direcção de movimento desejada está algum objecto sólido ou não. Caso esteja, o sistema de colisão é activado e a personagem principal é impedida de se movimentar naquela direcção. O sistema de colisão, à semelhança do sistema de gravidade, também teve de ser cuidadosamente codificado e testado. A primeira tentativa de implementação deste sistema revelou bastantes incoerências, nomeadamente, a mais séria, a personagem principal ficar presa em algumas plataformas obrigando o utilizador a reiniciar o jogo. Deste modo, a solução para este problema passou por modificar o sistema até então implementado para verificar ao mesmo tempo se existe alguma colisão na horizontal e vertical em todos os sentidos possíveis. Esta solução foi implementada graças à ajuda encontrada após alguma pesquisa realizada na Internet;
- Uma verificação para controlar a barra de vida restante à personagem principal. Esta vai diminuíndo à medida que a personagem principal vai sendo atingida pelos inúmeros inimigos e, assim que o valor de vida restante atinge zero, a personagem principal morre e o nível recomeça do início. O valor de vida restante pode aumentar caso a personagem principal entre em contacto com o coração (obj_Heart). Contudo, colocar a barra de vida restante no ecrã de nível levantou alguns problemas, visto que o modo de colocação utilizado, por exemplo, em qualquer plataforma existente em qualquer dos níveis existentes neste jogo não seria suficiente, na medida em que este jogo faz uso de uma funcionalidade incluída na ferramenta Game Maker Studio denominada de "View", funcionalidade essa responsável por manter sempre centrada a camera na personagem principal, e, sendo assim, o modo de colocação utilizado foi o modo dinâmico, isto é, sempre que a "View" rola para a esquerda ou para a direita, a posição da barra de vida restante é calculada e, em seguida, actualizada para simular que esta se mantém inalterada no mesmo lugar, quando no fundo acontece é esta ser redesenhada novamente na nova posição em que a "View" se encontra. A solução para este problema foi igualmente encontrada após alguma pesquisa na Internet;
- Uma verificação para controlar a barra de pontos de experiência ganhos a eliminar os diversos inimigos e, à medida que esses pontos de experiência vão subindo, a personagem principal vai subindo de nível. A colocação quer da barra de pontos de experiência, quer do nível da personagem principal ou quer do contador de pontos acumulados no ecrã levantou os mesmos problemas que a colocação da barra de vida restante, já explicada anteriormente neste relatório, levantou.

Para além destas funcionalidades implementadas mais importantes para o funcionamento do jogo, existem outras que embora não sejam tão importantes quanto as referidas anteriormente, também devem ser mencionadas neste relatório:

- Teste de colisão entre a personagem principal e o objecto que permite passar ao próximo nível;
- Teste de colisão entre a personagem principal e os dois tipos de gemas existentes;
- Teste de colisão entre a personagem principal e os dois tipos de molas existentes com especial atenção ao facto de as molas apenas funcionarem caso a personagem principal esteja directamente acima delas.

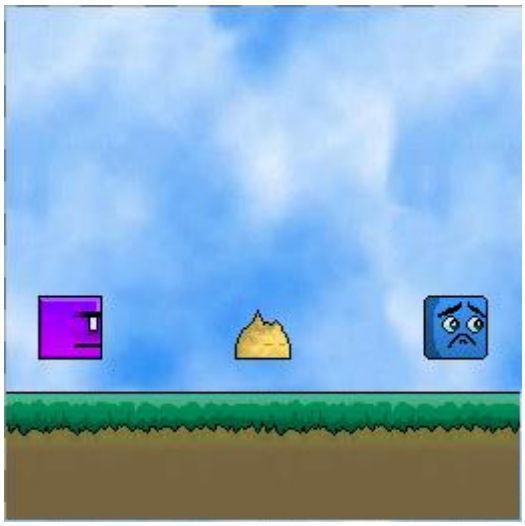
Por último, para finalizar a explicação detalhada deste objecto, falta apenas referir que mediante o que vai acontecendo à personagem principal, como por exemplo, movimentar-se para a esquerda ou para a direita ou saltar, esta vai alterando o seu aspecto, sendo que, neste caso em particular, a direcção da seta presente no centro da imagem do objecto Personagem Principal varia consoante a direcção do movimento do mesmo.

Coleccionáveis:



Tal como já foi referido, existem três tipos de objectos diferentes que a personagem principal pode colectar. São eles: gema azul (obj_gem), gema roxa (obj_SquareGem) e coração (obj_Heart). Os dois tipos de gemas diferem na pontuação atribuída ao apanhar cada uma delas. A gema azul aumenta a pontuação em 1000 pontos a gema roxa aumenta a pontuação em 2500 pontos. Quanto ao coração, este aumenta o valor de vida restante em 75 e aumenta a pontuação em 250 pontos.

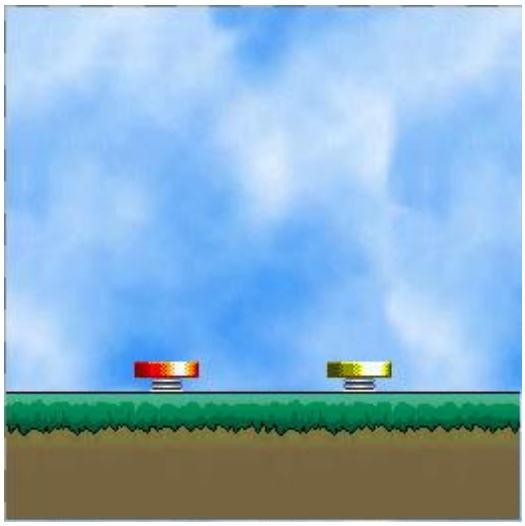
Inimigos:



Existem três tipos diferentes de inimigos: o inimigo roxo (obj_enemy), o inimigo creme (obj_enemy2) e o inimigo azul (obj_enemy3). Cada inimigo tem um comportamento diferente: o inimigo roxo é mais o agressivo dos três, na medida em que vagueia pelo mapa e persegue assim que detectada a personagem principal causando-lhe um dano médio; o inimigo creme é o que causa mais dano dos três, fica imóvel até detectar a personagem principal começando também a persegui-la tal como acontece com o inimigo roxo; por último, o inimigo azul é o mais pacífico dos três servindo apenas para aumentar a pontuação, visto que assim que a personagem principal é detectada, este foge na direcção oposta. Estes três inimigos também partilham grande parte das verificações descritas anteriormente no segmento

referente à personagem principal. Os problemas mais relevantes encontrados foram: primeiro, a codificação da inteligência artificial de todos os inimigos, visto que estes teriam de ser capazes de localizar a personagem principal e, em seguida, atacá-la enquanto navegam pelos diversos obstáculos existentes nos cinco mapas e, segundo, a colocação do valor de vida restante imediatamente acima de todos os inimigos. A solução para o primeiro problema, descrito anteriormente, foi encontrada após alguma pesquisa na Internet. Quanto à solução para o segundo problema, também descrito anteriormente, foi implementada com base no que foi implementado para a colocação da barra de vida restante da personagem principal, já devidamente explicada anteriormente neste relatório, com a pequena diferença de que em vez de actualizar a posição do objecto em questão à medida que a "View" é alterada, a posição do objecto em questão é actualizada à medida que todos os inimigos se movimentam.

Molas:



Existem dois tipos de molas: a mola vermelha e a molha amarela.A função das molas no jogo é a de quando a personagem principal entra em contacto com a parte superior da uma qualquer mola, esta altera momentaneamente o salto da personagem principal lançando esta a uma altura média no caso da mola vermelha ou a uma altura elevada no caso da mola amarela.

4. Resultados

Após dias, semanas, meses de trabalho, foi finalmente atingido o objectivo deste trabalho: desenvolver um jogo Windows 2D de plataformas. A versão final do jogo pode assim ser instalada em qualquer computador Windows com recurso ao executável de instalação do jogo.

Durante o processo de desenvolvimento, foram produzidas algumas versões inacabadas do jogo e realizados pequenos testes a alguns utilizadores para observar as suas reacções. Algumas reacções foram positivas outras nem tanto, mas todos estes testes durante o desenvolvimento do jogo foram de extrema importância para chegar ao resultado final.

Estando a versão final do jogo terminada, foram realizados novos testes junto de um grupo de entusiastas dos videojogos para que estes pudessem dar a sua opinião sobre a versão final do jogo. A opinião foi unânime, o jogo faz lembrar sem sombra de dúvida jogos clássicos como Donkey Kong, Super Mario Bros ou Sonic e proporciona uma experiência agradável, desafiante e divertida ao utilizador.

5. Conclusões e Trabalho Futuro

Para concluir este relatório, todo o processo de desenvolvimento deste trabalho levou práticamente todo o semestre para ser terminado. Foram retidos inúmeros aspectos e conceitos da utilização da ferramenta Game Maker Studio e da linguagem GML possívelmente muito importantes para o futuro e foi concluído que a ferramenta escolhida apresenta foi ideal, na medida em que permitiu cumprir o objectivo deste trabalho.

O jogo, depois de terminado, cumpre as espectativas, é sem qualquer dúvida um jogo de plataformas baseado em jogos como Donkey Kong, Super Mario Bros e Sonic. Apersonagem principal tem por objectivo atravessar cada um níveis existentes enfrentando os diversos inimigos e inúmeros obstáculos no seu caminho tendo por objectivo completar todos os níveis com o maior número de pontos possível. Contudo, como ajuda, a personagem principal tem vários items que pode apanhar ao longo de todo o jogo.

Relativamente ao trabalho futuro, as possibilidades são perto de ilimitadas. Em primeiro lugar, é possível com algumas modificações no código fonte do jogo a migração para as plataformas móveis Android e iOS. De seguida, para dinamizar ainda mais o jogo, criar um modo multijogador, no mesmo computador e/ou pela Internet, de forma a tornar o jogo mais sociável. Por último, estando desenvolvido o modo multijogador, o caminho a seguir seria certamente possibilitar interacção com as diversas redes sociais como é o caso do Facebook permitindo aos utilizadores partilhar dicas, pontuações, truques, entre outros.

Bibliografia

- Alexandre Pereira, Carlos Poupa (2004, Pereira). Como escrever uma Tese, monografia ou livro científico usando o Word, 3ª edição, Lisboa: Edições Sílabo;
- Tutoriais e referências a código fonte utilizados podem ser encontrados em: http://wiki.yoyogames.com/index.php/Main_Page
- Outros tutoriais podem ser encontradas em: http://www.youtube.com
- Pacotes de recursos para Game Maker Studio utilizados podem ser encontrados em: http://sandbox.yoyogames.com/make/resources
- Outros podem ser encontrados em: http://www.spriters-resource.com e http://themushroomkingdom.net/wav.shtml
- Exemplos e ideias para o jogo podem ser encontrados em: http://sandbox.yoyogames.com/browse?category=4

Cronograma

Através do quadro seguinte, é possível observar o tempo despendido para cada uma das diversas tarefas realizadas, sendo representado pelas semanas do referido mês.

Tarefa	Fevereiro	Março	Abril	Maio	Junho	Julho
Submissão de Proposta de Trabalho Final de Curso	3ª e 4ª					
Planeamento da Estrutura e Lista de Requisitos do Projecto		1ª, 2ª, 3ª e 4ª				
Escolha da Ferramenta a Utilizar		4ª	1ª	3ª e 4ª		
Implementação do Ecrã de Menu Principal					4ª	1ª
Implementação do Ecrã de Fim					4ª	1ª
Implementação do Ecrã de Nível					1ª, 2ª, 3ª e 4ª	1ª e 2ª
Implementação das Funcionalidades Avançadas					1ª, 2ª, 3ª e 4ª	1ª e 2ª
Relatório Final						3ª

Anexos

Anexo A – Personagem Principal

Information about object: obj_player
Sprite: spr_right_still
Solid: false
Visible: true
Depth: -500
Persistent: true
Parent:
Mask: spr_collision_mask
Create Event:
execute code:

```
// Variables \\
came_from="";cantbehit=false;trans86E=false;dead=false;current_room=room;visibl=f
alse;hit=false;right="Right";left="Left";nowhere="Nowhere";up="Up";down="Down";
going=nowhere;continuesprite=-1;keysright=false;keysleft=false;freespace="Free
space";liquid="Liquid";in=freespace;inwater=false;hitshock=false;going_to_next_room
=false;

facing=left;
speedx=0;
speedxmax=4;
slip=8;
startup=11;
grav=0.6;
move=0;

keyleft=vk_left;
keyright=vk_right;
keyup=vk_space;
keydown=vk_down;

onground=false;
vspeedmax=12;
jumpspeed=11;

playerscore=0;

invins_max=90;
invins=0;

    // Water Variables \\
    speeddivide=2;
    vspeeddivide=2;
    hopout=1.2;
```

```
   gravdivide=2;
   jumpspeeddivide=1.7;
   maxvspeeddivide=10;

// Camera \\
instance_create(x,y,CAMERA);
view_object=CAMERA;
instance_create(x,y,ScoreOperator);

// Exp And Hp \\
hpmax=100;
hp=hpmax;
hpscale=150;

expmax=100;
expcurrent=0;
expscale=150;

level=1;
End Step Event:
execute code:

// Player Dies \\
if (hp==0&&dead=true)
{
persistent=true;
if (came_from=left)
   {x=obj_spawn_right.x; y=obj_spawn_right.y; hp=hpmax/2; dead=false;
audio_play_sound(sound_KillCharacter,true,0);}
if (came_from="")
   {x=obj_first_spawn.x; y=obj_first_spawn.y; hp=hpmax/2; dead=false;
audio_play_sound(sound_KillCharacter,true,0);}
}

// Main \\
if (hp<=0 ){dead=true;}

if (place_free(x,y+1)) {gravity=grav;}
else {gravity=0;}
if (vspeed>vspeedmax) {vspeed=vspeedmax;}

if (keyboard_check_direct(keyright)&&keysright==false) {move=1; keysright=true;}
if (keyboard_check_direct(keyleft )&&keysleft==false)  {move=2; keysleft=true}
if (keyboard_check_direct(keyright)&&!keyboard_check_direct(keyleft )) {move=1;}
if (keyboard_check_direct(keyleft )&&!keyboard_check_direct(keyright)) {move=2;}
if (!keyboard_check_direct(keyright)&&keysright==true) {move=0; keysright=false;}
if (!keyboard_check_direct(keyleft )&&keysleft==true)  {move=0; keysleft=false;}

if (!hit)
{if (move==1)          {speedx+=(speedxmax-speedx)/startup}
```

```
else if (move==2)       {speedx+=(-speedxmax-speedx)/startup}
if (trans86E=false)
{if (move==0&&speedx!=0) {speedx+=-(speedx/slip)}}
if (speedx>-0.3&&speedx<0.3&&move=0) {speedx=0;}}
if speedx>speedxmax {speedx=speedxmax;}
else if speedx<-speedxmax {speedx=-speedxmax;}

if (keyboard_check_pressed(keyup))
{
if (!place_free(x,y+1)&&place_free(x,y-1))||(inwater)
{
vspeed=-jumpspeed; audio_play_sound(sound_Jump,false,0); if (inwater)
{if (facing==right)  {sprite_index=spr_right_swim; image_index=0; image_speed=0.5;}
if (facing==left)   {sprite_index=spr_left_swim;  image_index=0;
image_speed=0.5;}}}}

if (keyboard_check_released(keyup)&&going==up) {vspeed/=1.6;}

if (speedx>0&&continuesprite=-1&&!hitshock) {facing=right;}
if (speedx<0&&continuesprite=-1&&!hitshock) {facing=left;}
if (!place_free(x,y+1)) {onground=true;}
else {onground=false;}
if (vspeed>0) {going=down;}
else if (vspeed<0) {going=up;}
else {going=nowhere;}

if (place_meeting(x,y,water)&&inwater==false)
{
   in=liquid; inwater=true;
   speedxmax/=speeddivide;
   vspeed/=vspeeddivide;
   grav/=gravdivide;
   jumpspeed/=jumpspeeddivide;
   vspeedmax/=maxvspeeddivide;
   audio_play_sound(sound_Splash,true,0);
   instance_create(x+speedx,y,obj_splash);
}

else if (!place_meeting(x,y,water)&&inwater==true)
{
   in=freespace; inwater=false;
   speedxmax*=speeddivide;
   vspeed*=hopout;
   grav*=gravdivide
   jumpspeed*=jumpspeeddivide;
   vspeedmax*=maxvspeeddivide;
}

if (speedx==0&&onground&&move=0&&!hitshock)
{
```

```
if (facing==right) {sprite_index=spr_right_still;    image_speed=0.25;}
else if (facing==left) {sprite_index=spr_left_still; image_speed=0.25;}
}

if (speedx!=0&&onground&&!hitshock)
{
if (facing==right)     {sprite_index=spr_right; image_speed=0.5;}
else if (facing==left) {sprite_index=spr_left;  image_speed=0.5;}
if (facing==right&&move==2&&continuesprite==-1&&abs(speedx)>speedxmax/1.8)
{sprite_index=spr_right_toleft; image_speed=0.5; facing=left ;
continuesprite=sprite_index; image_index=0;}
else if (facing==left&&move==1&&continuesprite==-
1&&abs(speedx)>speedxmax/1.8)
{sprite_index=spr_left_toright; image_speed=0.5; facing=right;
continuesprite=sprite_index; image_index=0;}
}

if (!onground&&!inwater&&!hitshock)
{
if     (facing==right&&going==up)   {sprite_index=spr_right_up;   image_speed=0;}
else if (facing==right&&going==down) {sprite_index=spr_right_down;
image_speed=0;}
if     (facing==left&&going==up)    {sprite_index=spr_left_up;    image_speed=0;}
else if (facing==left&&going==down)  {sprite_index=spr_left_down;
image_speed=0;}
}

if (!onground&&inwater&&!hitshock)
{
if (facing==left&&sprite_index==spr_right_swim)       {sprite_index=spr_left_swim;
image_speed=0.5;}
else if (facing==right&&sprite_index==spr_left_swim)  {sprite_index=spr_right_swim;
image_speed=0.5;}
if (facing==right&&sprite_index!=spr_right_swim)      {sprite_index=spr_right_water;
image_speed=0.5;}
else if (facing==left&&sprite_index!=spr_left_swim)   {sprite_index=spr_left_water;
image_speed=0.5;}
}

if (!onground) {continuesprite=-1;}
else if (facing==right&&move==2) {continuesprite=-1;}
else if (facing==left&&move==1) {continuesprite=-1;}
if (continuesprite>-1)
{sprite_index=continuesprite; image_speed=0.5;
if (facing==left) {if speedx<0 {speedx=0;}}
if (facing==right) {if speedx>0 {speedx=0;}}}

if (place_free(x+speedx,y)) {x+=speedx;}
else
{
```

```
if (trans86E=false)
{
if (speedx>0) {move_contact_solid(0  , speedx); speedx=0;}
if (speedx<0) {move_contact_solid(180,-speedx); speedx=0;}
}
}

// Player After Taking Damage \\
if (going==down or going==nowhere) {cantbehit=false;}

hitshock=hit
if (hit&&invins=0)
{
if (facing==right) {sprite_index=spr_right_knockback; image_index=0;
image_speed=0.5;}
if (facing==left) {sprite_index=spr_left_knockback; image_index=0;
image_speed=0.5;}
vspeed=-4;
invins=invins_max;
}
else if (!place_free(x,y+1)) {hit=false;}
else if (inwater&&vspeed>0) {hit=false;}

if (invins>0&&visibl==true) {visibl=false;}
else if (invins>0&&visibl==false) {visibl=true;}

if (invins>0) {invins-=1;}
if (invins==0) {visibl=true;}

if (hp>hpmax) {hp=hpmax;}
if (hp<0) {hp=0;}

if (!place_free(x,y+1)) {canbehit=false;}

// Player Level Up \\
if (expcurrent>=expmax)
{
for (i=0; i>-1; i+=1)
{
if expcurrent>=expmax
{
   expcurrent-=expmax;
   level+=1;

   expmax+=round((expmax/1+(level/32))/(3.5+expmax/1500))

   hpmax += 10;
   hp=hpmax;

   audio_play_sound(sound_LevelUp,true,0);
```

```
}
if expcurrent< current_room="room;" {y-="1;}}" (!place_free(x,y)) while
audio_play_sound(sound_LevelCompleted,true,0); trans86E="false;" y="room_height;"
x="obj_spawn_right.x;" {going_to_next_room="false;"
(going_to_next_room&&current_room!="room)" if else
instance_create(view_xview,view_yview,room_transition);} {room_goto_next();
(going_to_next_room&&current_room="room)" \\ Changing Room } {i="-4;}">
Collision Event with object obj_next_room:
execute code:

going_to_next_room=true;
trans86E=true;
came_from=left;
Collision Event with object obj_gem:
execute code:

playerscore+=1000;
audio_play_sound(sound_Gem,true,0);
with (other) instance_destroy();
Collision Event with object obj_SquareGem:
execute code:

playerscore+=2500;
audio_play_sound(sound_SquareGem,true,0);
with (other) instance_destroy();
Collision Event with object obj_Heart:
execute code:

hp+=75;
playerscore+=250;
audio_play_sound(sound_Heart,true,0);
with (other) instance_destroy();
Collision Event with object obj_floor:
execute code:

y=round(y)
move_contact_solid(direction,abs(vspeedmax)+1);
vspeed=0;
Collision Event with object obj_enemy:
execute code:

if
(vspeed>=0&&place_free(x,y+1)&&place_meeting(x,y+vspeed,other)&&!hit&&!keyb
oard_check(keydown)&&!inwater) {
   other.hp-=2;
   vspeed=-8;
   other.vspeed+=4;
   cantbehit=true;

   playerscore+=100;
```

```
   audio_play_sound(sound_HurtMonster,true,0);
}
else if (vspeed<=0) or (keyboard_check(keydown) or (inwater))
{
if (!hit&&invins=0&&!cantbehit)
{
hit=true;
hp-=other.strength;

audio_play_sound(sound_HurtCharacter,true,0);
}
}
```
Collision Event with object obj_Spring1:
execute code:

```
if place_meeting(x, y + vspeed, obj_Spring1)
{
   while (!place_meeting(x, y + sign(vspeed), obj_Spring1)) y += sign(vspeed);
   vspeed=-15;
   audio_play_sound(sound_Spring,true,0);
}
```

Collision Event with object obj_Spring2:
execute code:

```
if place_meeting(x, y + vspeed, obj_Spring2)
{
   while (!place_meeting(x, y + sign(vspeed), obj_Spring2)) y += sign(vspeed);
   vspeed=-19;
   audio_play_sound(sound_Spring,true,0);
}
```

Other Event: Animation End:
execute code:

```
if (sprite_index==continuesprite) {continuesprite=-1; image_index=image_number-1;
speedx=0;}
if (sprite_index==spr_right_swim)      {sprite_index=spr_right_water; image_index=0;
}
else if (sprite_index==spr_left_swim)  {sprite_index=spr_left_water; image_index=0; }

if (sprite_index==spr_right_knockback)||(sprite_index==spr_left_knockback)
{image_index=image_number-1; image_speed=0;}
```
Draw Event:
execute code:

```
if (visibl==true) {draw_sprite(sprite_index,floor(image_index),x,y);}

draw_sprite(spr_hp_bar,0,view_xview+3,view_yview+3);
```

```
draw_sprite(spr_exp_bar,0,view_xview+3,view_yview+39);
draw_sprite_stretched(spr_hp_fill,0,view_xview+54,view_yview+12,hp/hpmax*hpscal
e,14);
draw_sprite_stretched(spr_exp_fill,0,view_xview+54,view_yview+39+9,expcurrent/ex
pmax*expscale,14);

draw_set_font(Health);
draw_set_halign(true)
draw_text((view_xview+54)+hpscale/2,view_yview+11,string(hp));
draw_text((view_xview+54)+expscale/2,view_yview+39+8,string(expcurrent));
draw_set_halign(false)
draw_set_color(c_white);
```

Anexo B – Botão "Play"

Information about object: START
Sprite: spr_startbutton
Solid: false
Visible: true
Depth: 0
Persistent: false
Parent:
Mask:
Create Event:
execute code:

```
image_speed=0;
```

End Step Event:
execute code:

```
if (mouse_check_button(mb_left)) {if image_index=1 {image_index=2;}}
```

Mouse Event for Left Pressed:
execute code:

```
image_index=2;
```

Mouse Event for Left Released:
execute code:

```
room_goto_next();
```

Mouse Event for Mouse Enter:
execute code:

```
image_index=1;
Mouse Event for Mouse Leave:
execute code:

image_index=0;
```

Anexo C – Botão "End"

Information about object: END
Sprite: spr_backbutton
Solid: false
Visible: true
Depth: 0
Persistent: false
Parent:
Mask:
Mouse Event for Left Released:
execute code:

```
with (obj_player)
{
   instance_destroy();
}
with (CAMERA)
{
   instance_destroy();
}
with (ScoreOperator)
```

```
{
   instance_destroy();
}
room_goto(rm_MainMenu);
```

Anexo D – Inimigo Roxo

Information about object: obj_enemy
Sprite: spr_e_right_idle
Solid: false
Visible: true
Depth: 0
Persistent: false
Parent:
Mask:
Create Event:
execute code:

```
Left_Sight=   160;
Right_Sight=  160;
Above_Sight=  64;
Below_Sight=  32;
```

```
strength=10;

right_walk=spr_e_right     ;
right_die=spr_e_right_dying;
right_idle=spr_e_right_idle;
right_up=spr_e_right_up    ;
right_down=spr_e_right_down;

left_walk=spr_e_left     ;
left_die=spr_e_left_dying;
left_idle=spr_e_left_idle;
left_up=spr_e_left_up    ;
left_down=spr_e_left_down;

forget_him_time=45;

before_next_jump_set=20;
before_next_jump=before_next_jump_set;

forget_him=0;jump=false;found_him=false;right="Right";left="Left";nowhere="Nowhe
re";up="Up";down="Down";going=nowhere;continuesprite=-1;freespace="Free
space";liquid="Liquid";in=freespace;inwater=false;speedxmax_plus=0;

facing=left;
speedx=0;
speedxmax=1;
slip=8;
startup=11;
grav=0.6;
move=choose(1,2);
onground=false;
vspeedmax=12;
jumpspeed=9;

    // Water Variables \\
    speeddivide=2;
    vspeeddivide=2;
    hopout=1.2;
    gravdivide=2;
    jumpspeeddivide=1.7;
    maxvspeeddivide=10;

hpmax=10;
hp=hpmax;
Destroy Event:
execute code:

obj_player.expcurrent+=42;
End Step Event:
```

execute code:

```
if (place_free(x,y+1)) {gravity=grav;}
else {gravity=0;}
if (vspeed>vspeedmax) {vspeed=vspeedmax;}
if (hp>0)
{
if (move==1) {speedx+=((speedxmax+speedxmax_plus)-speedx)/startup}
else if (move==2) {speedx+=(-(speedxmax+speedxmax_plus)-speedx)/startup}
else if (move==0&&speedx!=0) {speedx+=-(speedx/slip)}
if (speedx>-0.3&&speedx<0.3&&move=0) {speedx=0;}
if speedx>(speedxmax+speedxmax_plus) {speedx=(speedxmax+speedxmax_plus);}
else if speedx<-(speedxmax+speedxmax_plus) {speedx=-
(speedxmax+speedxmax_plus);}
if (place_meeting(x,y,water)&&inwater==false)
{in=liquid; inwater=true;
speedxmax/=speeddivide;
vspeed/=vspeeddivide;
grav/=gravdivide;
jumpspeed/=jumpspeeddivide;
vspeedmax/=maxvspeeddivide;
instance_create(x+speedx,y,obj_splash);}
else if (!place_meeting(x,y,water)&&inwater==true)
{in=freespace; inwater=false;
speedxmax*=speeddivide;
vspeed*=hopout;
grav*=gravdivide
jumpspeed*=jumpspeeddivide;
vspeedmax*=maxvspeeddivide;}
if (speedx>0) {facing=right;}
if (speedx<0) {facing=left;}
if (!place_free(x,y+1)) {onground=true;}else {onground=false;}
if (vspeed>0) {going=down;}
else if (vspeed<0) {going=up;}else {going=nowhere;}
if before_next_jump>0 {before_next_jump-=1;}
if (jump=true&&before_next_jump=0)
{jump=false;
before_next_jump=before_next_jump_set+floor(random(20));
if (!place_free(x,y+1)&&place_free(x,y-1))||(inwater)
{vspeed=-jumpspeed;}}

if (speedx==0&&onground&&move=0)
{if (facing==right) {sprite_index=right_idle;    image_speed=0.25;}
else if (facing==left) {sprite_index=left_idle; image_speed=0.25;}}

if (speedx!=0&&onground)
{if (facing==right)     {sprite_index=right_walk; image_speed=0.5;}
else if (facing==left) {sprite_index=left_walk;  image_speed=0.5;}
}
```

```
if (!onground)
{if     (facing==right&&going==up)   {sprite_index=right_up;   image_speed=0;}
else if  (facing==right&&going==down) {sprite_index=right_down; image_speed=0;}
if      (facing==left&&going==up)    {sprite_index=left_up;    image_speed=0;}
else if  (facing==left&&going==down)  {sprite_index=left_down;  image_speed=0;}}

if
(collision_rectangle(x-Left_Sight,y-
Above_Sight,x+Right_Sight,y+Below_Sight,obj_player,false,true)
&&!collision_line(obj_player.x,obj_player.y,x,y,obj_floor,true,true))
{
found_him=true;
forget_him=forget_him_time;
}
else
{
if forget_him>0 {forget_him-=1;}
if forget_him=0 {found_him=false}
}}

if hp>hpmax {hp=hpmax;}
if hp<0 {hp=0;}

if (hp==0)
{
if (facing==right&&sprite_index!=right_die) {sprite_index=right_die; image_index=0;
image_speed=0.5;}
if (facing==left&&sprite_index!=left_die)  {sprite_index=left_die; image_index=0;
image_speed=0.5;}

audio_play_sound(sound_KillMonster,true,0);
}

if (hp>0)
{
if (found_him==true)
{
if     (obj_player.x>x&&floor(random(3))==1) {move=1;}
else if (obj_player.x<X&&FLOOR(RANDOM(3))==1) (speedx if { else
{x+="speedx;}" (place_free(x+speedx,y)) } speedxmax_plus="2;" {move="2;}"
(floor(random(500))="=1)" {jump="true;" (inwater) (floor(random(300))="=150)"
(!place_free(x-1,y)) before_next_jump="0;}" (floor(random(50))="=1)" {if
(!place_free(x+(speedx*5),y)) (!place_free(x+1,y)) (found_him="=false)"
(obj_player.y<y&&floor(random(60))="=1)">0) {move_contact_solid(0  , speedx);
speedx=0;}
if (speedx<0) {move_contact_solid(180,-speedx); speedx=0;}
}}
Collision Event with object obj_floor:
execute code:
```

```
y=round(y)
move_contact_solid(direction,abs(vspeedmax)+1);
vspeed=0;
```
Other Event: Animation End:
execute code:

```
if (sprite_index==right_die) {instance_destroy(); image_speed=0;
image_index=image_number-1;}
if (sprite_index==left_die ) {instance_destroy(); image_speed=0;
image_index=image_number-1;}
```
Draw Event:
execute code:

```
draw_sprite(sprite_index,floor(image_index),x,y);
draw_set_font(enemy_hp);
draw_set_halign(true);
draw_text(x,y-32,"HP: "+string(hp));
draw_set_halign(false);
```