Universidade Lusófona de Humanidades e Tecnolgias

Licenciatura de Engenharia Informática

Relatório de Trabalho Final de Curso

Trabalho Realizado:

Gun n Run

Top-Down Shooter

Aluno: **Fábio Alexandre Silva Mendes**

Docente: **Professor Sérgio Guerreiro**

Índice:

Resumo do trabalho

Este trabalho teve como objectivo realizar um shooter, onde o jogador tem que atravessar diversos níveis, por fim, chegando ao final do jogo. O jogador tem diferentes armas que pode usar como também uma variedade de inimigos que disparam contra ele ou o perseguem . O jogo tem um total de três níveis cada um com um certo número de sub-níveis. Existem diversas interacções entre o jogador e o ambiente em que se encontra no videojogo.

Abstract:

This project is a game for windows as a top-down shooter where the player has to cross various levels, reaching the end level and win the game. The game has different weapons and pickups and is confronted with different enemies. The game has a total of three levels each with their own sub-levels.

Introdução:

No tempo actual existe uma grande variedade de computadores (desktops) como portáteis, cuja sua utilização varia desde uso privado a uso público, trabalho ou lazer.

Tal utilização depende do utilizador que tem as suas necessidades e exigências, espectativas do computador que utiliza. Seja navegar na internet, usar o word ou jogar um videojogo.

O computador dá a possibilidade ao utilizador de jogar confortavelmente em casa, seja um jogo antigo ou um jogo topo de gama, fazendo uso do facto que em relação ás consolas é mais provável poder jogar um jogo de uma geração mais antiga. De facto os videojogos começaram no computador

Os videojogos pertencem a uma indústria já avançada e que está em constante evolução, devido ao facto que os videojogos evoluem em conjunto com a tecnologia.

Foi por estas razões que se decidiu desenvolver uma aplicação para windows dando origem ao videojogo.

Por isso foi decidido desenvolver um jogo para windows (top-down view).

Esta aplicação teve algumas influências do rpg Zelda (Nintendo), nomeadamente o desenho dos níveis e do shooter Contra (Konami). Através dessas influências foi possível criar uma ideia inovadora, envolvendo uma vista top-down com níveis ambos lineares e abertos para exploração tal como a energia frenética de um shooter. Portanto a inovação da aplicação é a sua natureza híbrida que retira de dois jogos, de dois géneros diferentes.

Enquadramento Teórico:

A ideia do projecto começou com o incentivo dos videojogos e a ideia de criar um. A decisão de ser uma aplicação windows derivou da experiência com jogos de computador (desktop e portáteis).

O projecto também se a associava ás cadeiras estudadas devido ao facto de ser programação orientada a objectos.

A primeira ferramenta a usar foi o XNA, após alguns testes foi decidido usar a ferramenta Game Maker Studio, apesar de dar a possibilidade de desenvolver aplicações para outras plataformas o objectivo continuou a ser para Windows.

Ferramenta do projecto:

Após feita a escolha do tema foi realizada uma pesquisa na *internet* para encontrar e decidir qual a ferramenta a utilizar no desenvolvimento do projecto.

Escolha:

Após alguma pesquisa deu-se à descoberta da ferramenta Game Maker Studio que pode ser encontrada na página

(http://www.yoyogames.com/gamemaker/studio)

Mas também foi sugerida a ferramenta XNA que se trata de uma ferramenta de desenvolvimento de videojogos da microsoft. Na página

(http://msdn.microsoft.com/en-us/centrum-xna.aspx)

Após a utilização das duas ferramentas para testes foi escolhida a ferramenta Game Maker Studio.

Esta escolha foi feita devido à facilidade de utilização da ferramenta e foi na mesma onde o projecto já estava num estado bastante desenvolvido em relação ao protótipo em XNA.

Em relação a recursos de apoio, tutorials, o Game Maker Studio tem uma ligeira vantagem em relação ao XNA, o Game Maker Studio tem uma vasta comunidade desde *developers* amadores a *developers* experientes, como consequência existe uma abundância de recursos nos forums da ferramenta como vídeos a explicar certas técnicas de programação.

Ferramenta Game Maker Studio:

O Game Maker Studio é uma ferramenta desenhada para o desenvolvimento de videojogos, lançada em 15 de Novembro de 1999, o seu desenvolvimento continua ainda na actualidade publicada pela Yo Yo Games.

Originalmente a ferramenta tem como objectivo dar a possibilidade de desenvolvimento básico que envolve um sistema grab and drop e desenvolvimento mais avançado através da linguagem da ferramenta baseada em scritps (*Game Maker Studio Language*).

-A ferramenta dispõe de certas funcionalidades automáticas que não necessitam da atenção programador tornando a sua utilização fácil e confortável.

-A execução do projecto desenvolvido é realizado internamente com todos os recursos nele incluído sem necessidade de emulador como android, por exemplo.

-Existe uma grande facilidade de introdução de componentes e elementos como sons, imagens, sprites.

Apesar das suas vantagens a sua natureza traz desvantagens. Como uma grande parte das funcionalidades são internas e automáticas a optimização do projecto sofre. Em termos de programação avançada a linguagem é uma certa desvantagem, sendo própria da ferramenta leva tempo a aprender e utilizar correctamente.

Métodos de desenvolvimento:

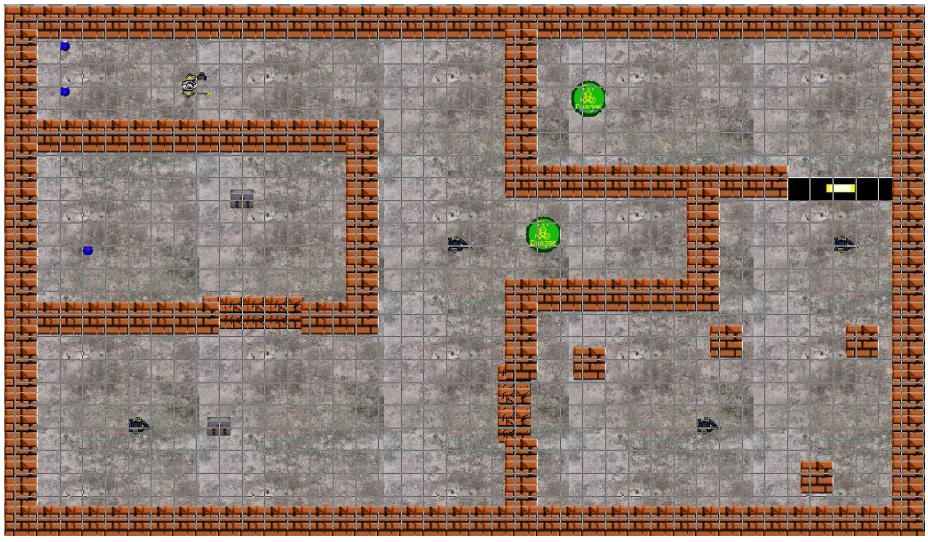
Menus:

1- Menu principal.
2- Menu fim de jogo (créditos).
3- Menu fim de jogo (Game Over).

Os menus são um meio de navegar no jogo, nomeadamente para iniciar e terminar o jogo. É através dos mesmos que fazem transições entre si.

Níveis:

Os níveis existentes são três compostos por várias áreas interligadas. A distinção pode ser feita através dos ambientes que representam. Os ambientes ou temas são uma base (primeiro nível), uma floresta (segundo nível) e uma montanha (terceiro nível). Representado na imagem seguinte:



Os designados sub-níveis são divididos em salas cada uma com o seu estilo de desenho. A transição entre níveis e as suas respectivas áreas é realizada por um sistema de portais que levam o jogador para a próxima área e um sistema de warping, transportando o jogador para outra parte do nível, sala, quando sai da sua fronteira. Em relação á persistência o estado de cada sala é guardado que dá a possibilidade de backtracking se necessário.

Em certos casos no jogo é, de facto, necessário voltar atrás no nível para realizar certas acções para progredir ou adquirir algum pickup que ainda não tenha sido usado. As duas imagens exemplificam a transição:

Neste caso exemplifica que é possivél ser transportado entre duas zonas fechadas.

Player:

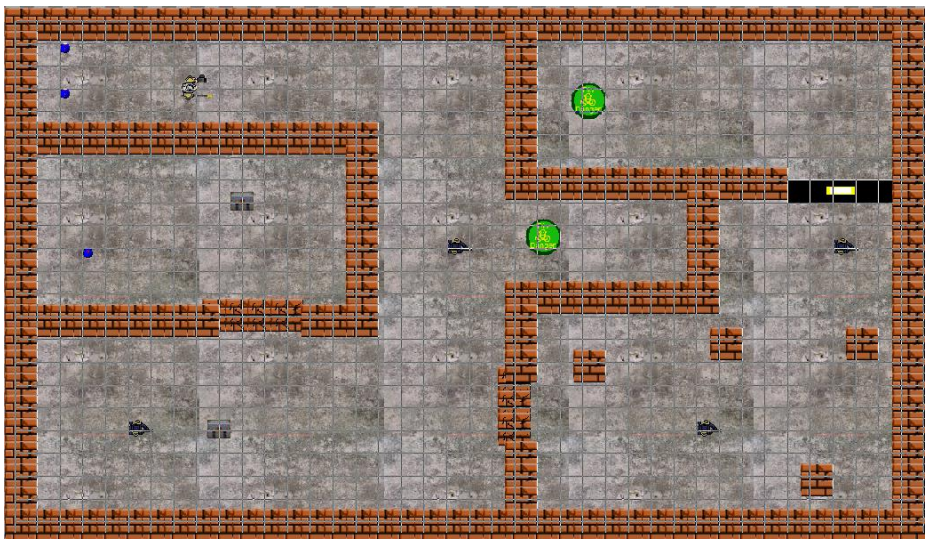
O designado player é o personagem que o utilizador controla no decorrer do jogo, é controlado por rato que tem a funcionalidade de apontar e disparar contra os inimigos enquanto o teclado é utilizado para movimentar o player com as teclas WASD, este formato é o formato standard de shooters.

Como qualquer jogo existe formas de utilizador "perder o jogo", quando sofre demasiado dano por parte de inimigos, seja por disparo ou colisão, morre e perde o jogo que mostra o menu Game Over com a opção de recomeçar ou sair do jogo.

Derrotar os inimigos mão é o único objectivo para o player. Existem chaves que se encontram nos níveis vitais para abrir as portas que são um obstáculo evidente. O player dispõe de diferentes armas que pode obter da mesma forma que as chaves.

A programação do player encontra-se no anexo A e C a E.

O player pode ser identificado no canto superior esquerdo da imagem:



Inimigos:

Estes objectos têm inteligência e agem dependendo do player, variam desde inimigos estacionários a inimigos que perseguem o player. Esta relação é realizada quando o inimigo identifica o player que entra no seu alcance, quando esta condição é satisfeita os inimigos disparam e perseguem o player. Quando o player consegue iludir os inimigos afastando-se do seu raio de alcance ficam estacionários até que identifiquem o player novamente.

Existe uma variedade de inimigos no jogo desenvolvido:

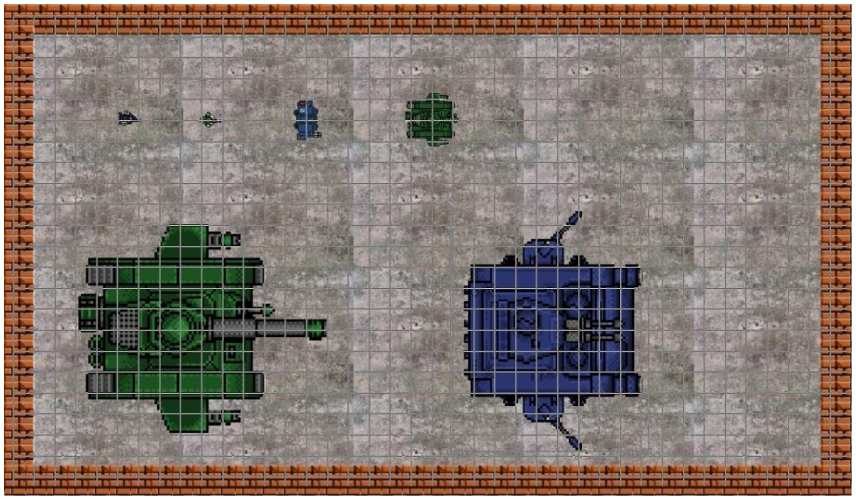
1-Um inimigo que segue o player até que este se afaste ou o destrua, se realizar contacto físico é destruído causando danos severos ao player. È o inimigo mais fraco do jogo. Este tipo de inimigo é comum em jogos cuja função é dar uma experiência de facilidade ao utilizador e dar conforto em relação ás mecânicas do jogo.

2-Um inimigo estacionário que dispara contra o player se entrar em alcance, este existem em três vertentes cuja dificuldade é maior em cada nível.

3-Os inimigos com maior dificuldade são inimigos que indicam o final do nível com a sua presença, em termos comuns são chamados de “boss”. Este tipo de inimigo é utilizado em quase todos os jogos, geralmente de tamanho desproporcional em relação aos restantes e maior dificuldade. Comparando com o primeiro inimigo mencionado o “boss” tem a função de dificultar o progresso ao player e causar pressão.

É esta diversidade que prende o utilizador ao jogo dá satisfação de completar o mesmo. Um jogo demasiado repetitivo não é apelativo.

A programação dos inimigos encontra-se no anexo B e F a K.



Aqui estão representados os inimigos de Gun n Run

Muralhas:

As muralhas são objectos inanimados com o objectivo de restringir a liberdade total do jogador de explorar o nível e torná-lo mais apelativo visualmente. O design das muralhas varia em relação ao tema de cada nível, um total de três. O seu design também é relativo ás suas propriedades de poder ou não destruir a muralha. Em certas zonas dos níveis é possível destruir parte das muralhas para dar acesso a lugares bloqueados.

Uma vez que o player entra em contacto com uma muralha não é possível mover ou atravessar a mesma seja destrutível ou não, o mesmo é aplicado aos inimigos. Projécteis não atravessam muralhas mas são a única maneira de destruir algumas muralhas. Esta funcionalidade de destruir muralhas nem sempre é evidente, isto leva ao utilizador explorar os níveis o que aprofunda a jogabilidade do jogo.

As muralhas também ajudam na dificuldade do jogo em certas partes limitando o espaço do player, é uma escolha de design por parte do designer e não uma mecânica de jogo.

A programação das interacções estão no anexo N.

Pickups:

Um dos aspectos mais importantes do jogo são os pickups, estes variam desde kits de vida, chaves e armas que ajudam o jogador a completar os níveis e tornar a jogabilidade mais apelativa. Os pickups não são só objectos mas também interacções com o jogador e com os inimigos indirectamente.

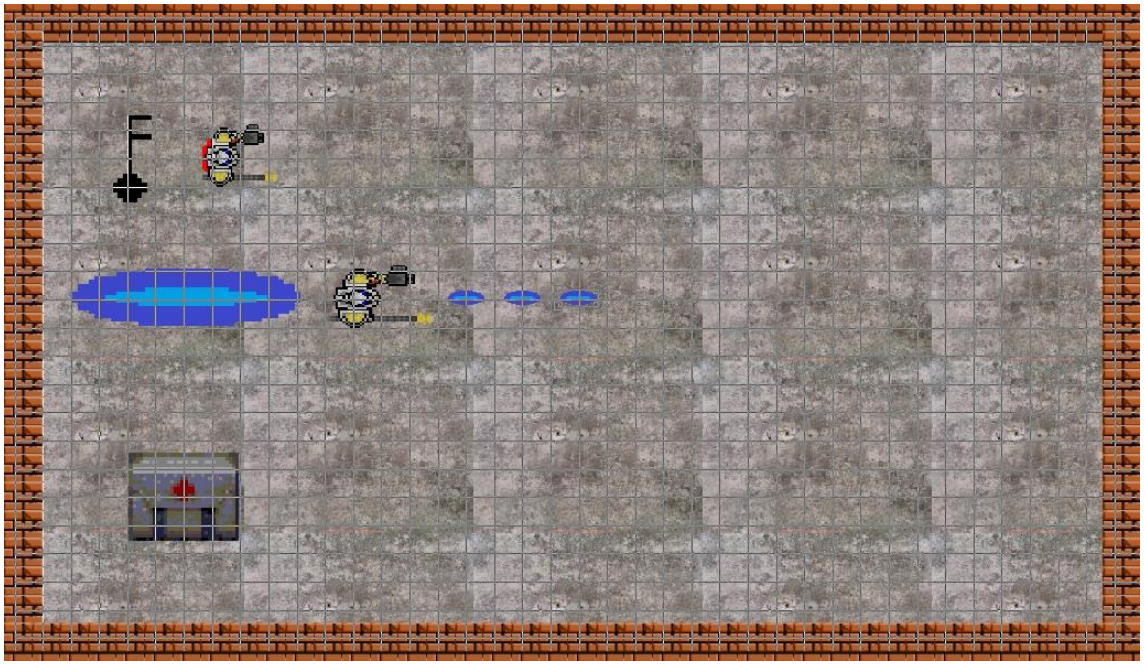
1-Kit de vida: Quando o jogador apanha um kit este restaura a sua vida. Esta é uma componente chave no jogo, faz com que o player se mantenha no jogo e torna a jogabilidade mais fácil e agradável.

2-Chave: Este é o único meio de abrir as portas nos diversos níveis. Quando um jogador apanha uma chave este volta à sua arma inicial não podendo apanhar outra arma até abrir a porta designada. No anexo l está referenciado como funciona. A programação da chave encontra-se no anexo L.

3-Arma laser: Uma das armas encontradas no jogo. É uma arma mais poderosa em relação á arma inicial com maior dano, alcance e velocidade. A programação do pickup da arma laser está no anexo O.

Estas interacções têm alguma complexidade, devido a regras impostas no desenvolvimento do jogo o jogador só pode ter um pickup com ele, enquanto não remover pickup actual não pode adquirir outro.

A imagem mostra os pickups de Gun n Run.



Portais e warps:

O sistema de portais e warps são o sistema de transição entre as diversas salas de cada nível.

Os portais são o que definem a linearidade do nível, só dispõem de um sentido como consequência não é possível voltar atrás. Existem excepções a esta regra por o simples facto de colocar dois portais a apontar para as duas salas.

Em um dos níveis existem elementos de free roam que significa poder andar livremente pelas salas, esta mecânica é realizada por warps. Warps são objectos invisíveis ao utilizador, com estes objectos é possível transitar entre salas e o seu estado é guardado,portanto, as salas ficam como o utilizador as deixou.

A transição do player é realizada através de uma relação que requer contacto físico para poder accionar a transição. Em termos técnicos existem variáveis no objecto que definem a posição do player e para que sala é transportado mas as variáveis são criadas na criação do objecto na sala. A programação dos portais encontra-se no anexo M

São estes objectos que dão amplitude aos níveis.

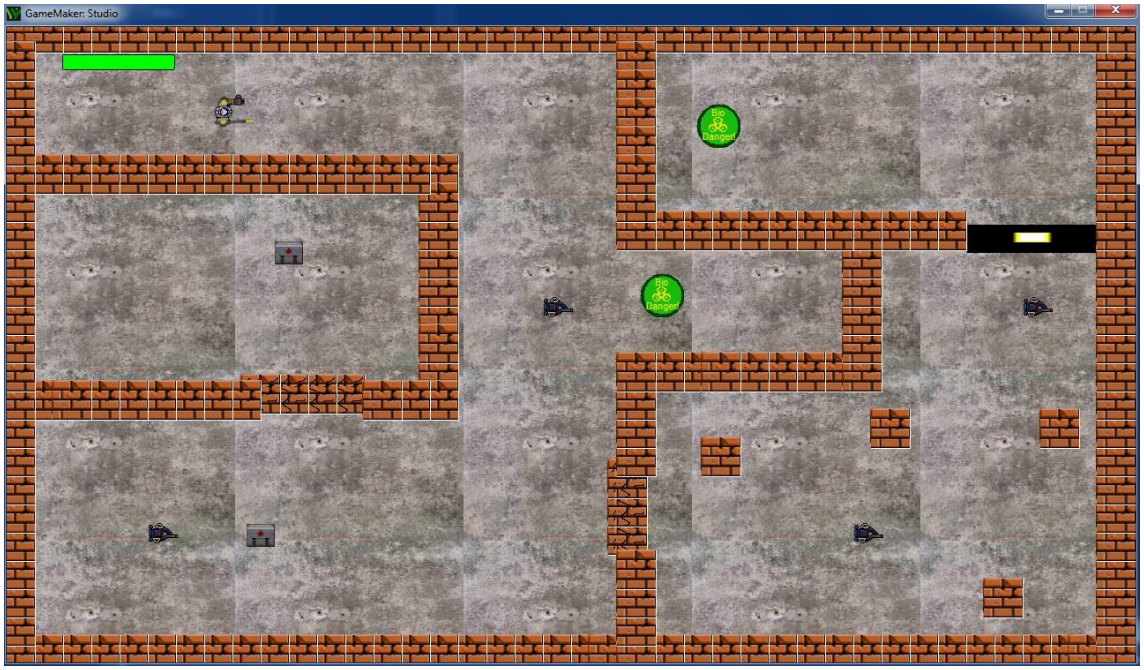
Barra de vida:

A barra de vida mostra a quantidade de hit points (hp) que o player tem, se a barra se esvaziar o player morre e o utilizador perde ,tal como, se a barra estiver cheia o player ainda não sofreu dano. A barra é essencial em qualquer jogo onde a morte não é instantânea, sem a barra era impossível verificar o estado da vida do player. Considerando a barra de vida como uma componente única, esta não define a vida do player é apenas uma informação.

Por outras palavras, a barra de vida faz parte de um sistema que cria, altera e mostra a vida do player. Podemos explicar este sistema de forma aprofundada.

O sistema de vida é implementado por dois controladores existentes nas salas, um que define a vida inicial e um responsável pelo desenho da barra de vida. Existem ainda mais uma variável que não existe nos controladores mas sim em cada projéctil dos inimigos, quando entram em contacto com o player a variável decrementa o valor da vida e termina o jogo quando esta for inferior ou igual a zero.

A programação da barra de vida está no anexo N.

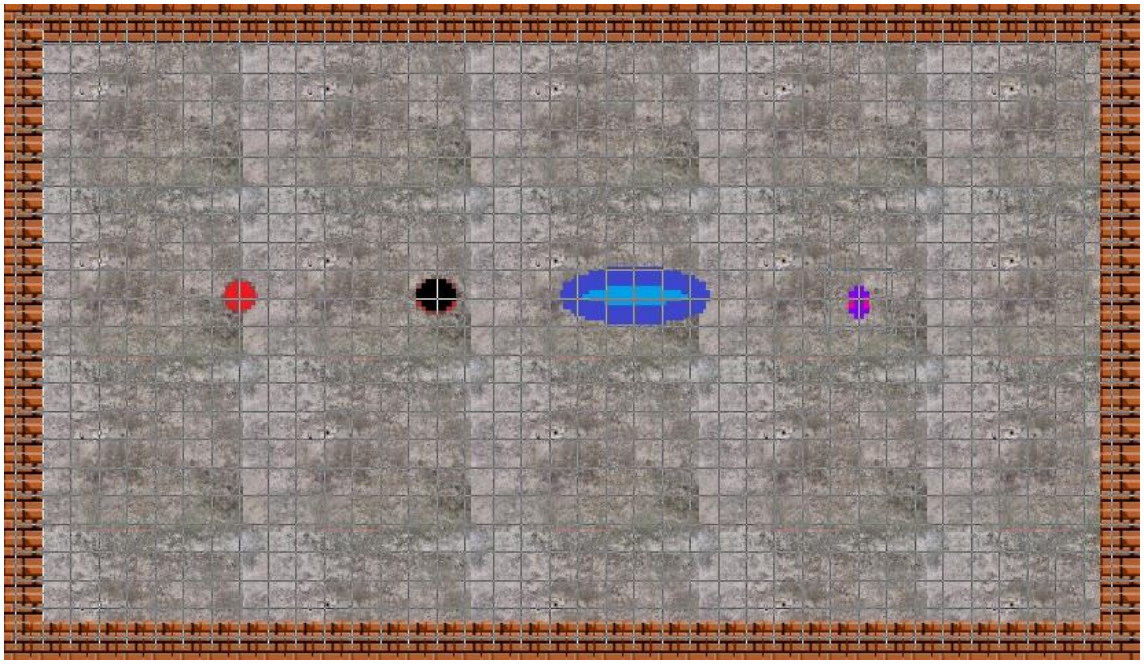


Projécteis:

Os únicos projécteis existentes em Gun n Run são as balas do player e dos inimigos com diferentes formas e percentagem de dano. Como referido anteriormente é nas balas que se decide o dano no player, a colisão da bala com o player especificamente.

A animação foi uma implementação complexa e não trivial, em relação aos inimigos o objecto tem que ser criado no sentido frontal de quem dispara e em direcção ao player sendo destruído quando entra em contacto com uma muralha. No que diz respeito ao player a implementação é idêntica com uma excepção, o objecto é criado no sentido do rato apontado pelo utilizador.

Existe código interno nos objectos que apenas executa em certas condições sendo a mais comum a colisão com outro objecto. A programação dos projécteis estão no anexo N.



Interacções:

A chave que faz com que o jogo desenvolvido funcione como pretendido são as interacções entre os vários objectos.

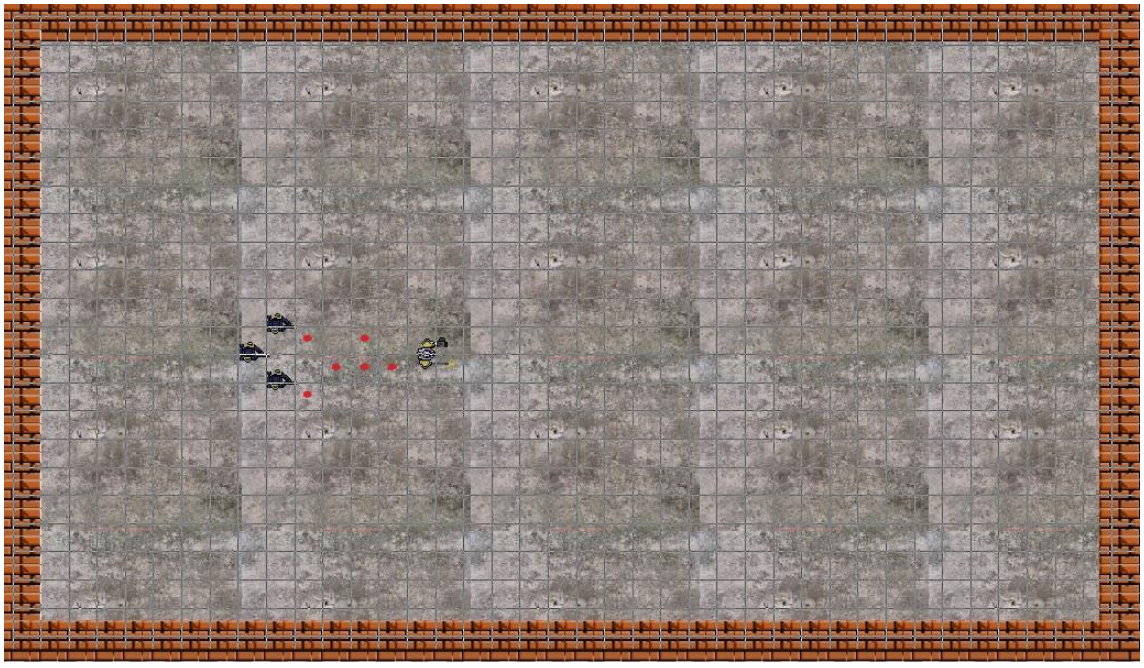
Para jogos deste tipo são necessárias no desenrolar da jogabilidade, os objectos agem conforme as acções do player , do ambiente, e outros objectos. Tal implementação é bastante complexa e árdua, talvez as interacções mais complicadas são as interacções com o player, em relação a inimigos programados já se tem conhecimento do seu comportamento mas em relação a uma personagem controlada por um ser humano, é necessário prever o comportamento imprevisível de um ser humano para restringir ou incentivar certas acções.

Existe uma ou mais interacções em cada objecto que executam certos eventos ou estados no jogo desenvolvido mas estão divididas em duas categorias:

1-Colisão: È uma interacção que existe quando há contacto físico entre objectos

2-Identificação: Um objecto identifica outro quando entram num raio de alcance

As interacções foram a parte mais intensiva do projecto e a que levou mais tempo a implementar. A programação das interacções estão no anexo N.



Extras:

Ouve uma fase do projecto onde o objectivo foi recolher recursos em diversas fontes para dar vida ao projecto.

As sprites são uma imagem ou um conjunto de imagens que dão fidelidade visual e gráfica aos objectos criados, existem alguns objectos que necessitam estar escondidos do utilizador e como consequência não têm sprites.

Os sons e música são recursos importantes pois dão mais realidade ao jogo e ajudam o utilizador a sentir-se em contacto com o jogo que está a jogar.

Processo de desenvolvimento:

Como em qualquer projecto desse tipo o seu desenvolvimento é demorado e complexo, o mesmo se verificou com a aplicação windows Gun n Run. O seu desenvolvimento apresentou alguns desafios em design e codificação.

A primeira dificuldade foi o facto de aprender a linguagem e a ferramenta em si, o Game Maker apresenta uma linguagem própria embora se baseie em objectos e a própria ferramenta tem certas funcionalidades que não são óbvias. Em termos de design a criação das salas apresentaram desafios inesperados, as salas têm de ser apelativas e balanceadas de modo a não prejudicar o utilizador. Houve alturas em que as salas eram difíceis de desenhar devido ao facto que por vezes eram muito triviais de completar ou muito complicadas para o jogador. A maior dificuldade do projecto foi a programação dos inimigos pois dar inteligência a objectos não é simples, os objectos têm que identificar e agir em relação aos outros objectos e ao player essas relações são a razão da dificuldade mencionada.

Em termos de programação do jogo é necessário e essencial criar interacções entre os objectos existentes, é uma fase cuja dificuldade reside no tempo gasto a identificar e implementar iterações, cada objecto pode conter uma ou várias interacções para um ou vários objectos e essas iterações têm que estar em sintonia para o jogo funcionar.

Resultados:

O projecto foi terminado após um longo desenvolvimento e planeamento, chegando ao seu estado final com todos os objectivos alcançados.

Foram realizados testes todos com resultados positivos, os utilizadores mostraram curiosidade e satisfação ao testarem a aplicação. Os testes foram feitos apenas em fase de desenvolvimento e não com a aplicação final.

A versão final da aplicação apresenta-se estável. Existem novos elementos que não se encontravam em versões anteriores, graças a testes efectuados foi possível implementar todas as funcionalidades pretendidas sem problemas. Os erros e bugs que estavam presentes na versão de desenvolvimento foram corrigidos. A aplicação dispõe de um instalador usando o windows MSI installer. As versões anteriores apresentavam um tamanho muito elevado ao que deveria ser (cerca de 130 MB), contudo, a aplicação sofreu uma compressão mas não apresenta qualquer impacto negativo sobre a mesma. Após a instalação da aplicação não é necessário ser iniciada constantemente pelo seu "exe", é possível repetir o jogo e facilmente se termina dentro da aplicação.

Conclusão:

O trabalho realizado foi um shooter, com vários níveis, chegando ao final do jogo. Existe uma diversidade de armas que pode usar como também uma variedade de inimigos que disparam contra ele ou o perseguem . O jogo tem um total de três níveis cada um com um certo número de sub-níveis.

Este projecto levou uma elevada porção de tempo a realizar e planear. Com este projecto foram adquiridos conhecimentos sobre a ferramenta e o desenvolvimento de videojogos. Levou a obrigação de estabelecer objectivos nas diversas fases de desenvolvimento.
Para concluir, este trabalho foi necessário uma grande quantidade de tempo para ser acabado.
Ao desenvolver este projecto o conhecimento sobre a ferramenta Game Maker Studio e a sua linguagem nativa GML foi adquirido e aprofundado.

Está planeado para o futuro melhorar o projecto tecnicamente e visualmente, implementar mais níveis e expandir para outras plataformas. Está planeado um maior leque de conteúdo com o objectivo de enriquecer o projecto.
Uma funcionalidade que pode ser implementada é um leaderboard que regista o melhor jogador a completar o jogo com maior número de pontos ou menor tempo.

Bibliografia:

Tutorials:

http://www.youtube.com/

Excertos de código:

http://www.yoyogames.com/gamemaker/studio

Sons e música:

http://gamebanana.com/

Sprites:

https://www.google.pt/webhp?hl=pt-PT&tab=ww

Anexos:

Anexo A-Movimento do jogador

Script movement;

```
if(keyboard_check(ord("W"))){
    y -= 7;
}else if(keyboard_check(ord("S"))){
    y += 7;
}
if(keyboard_check(ord("A"))){
    x -= 7;
}else if(keyboard_check(ord("D"))){
    x += 7;
}
```

Anexo B-Inteligência do inimigo:

Script detection_player;

```
if(life <= 0){
 sound_play(enemy_pain);
 instance_destroy();

}
if(distance_to_object(obj_player) <= 200){
        player_direction = point_direction(x,y,obj_player.x,obj_player.y);
        direction = player_direction;
        image_angle = player_direction;

    if(floor (random(20 - 1)) = 0){
        var obj;
        obj = instance_create(x,y,obj_enemy_bullet);
        with (obj) motion_set(point_direction(x,y,obj_player.x,obj_player.y),10);

        }

        action_potential_step(obj_player.x,obj_player.y,4,0);

}
```

Script detection_laser;

```
if(life <= 0){
 instance_destroy();
 sound_play(enemy_pain);
}

if(distance_to_object(obj_player_laser) <= 200){

      player_directionb = point_direction(x,y,obj_player_laser.x,obj_player_laser.y);
      obj_enemy.direction = player_directionb;
      obj_enemy.image_angle = player_directionb;

   if(floor (random(20 - 2)) = 0){
      var objlaser;
      objlaser = instance_create(x,y,obj_enemy_bullet);
      with (objlaser)
motion_set(point_direction(x,y,obj_player_laser.x,obj_player_laser.y),10);

      }

      action_potential_step(obj_player_laser.x,obj_player_laser.y,4,0);

}
```

Script detection_key;

```
if(life <= 0){
 instance_destroy();
 sound_play(enemy_pain);
}

 if(distance_to_object(obj_player_key) <= 200){

      player_directionc = point_direction(x,y,obj_player_key.x,obj_player_key.y);
      obj_enemy.direction = player_directionc;
      obj_enemy.image_angle = player_directionc;

   if(floor (random(20 - 3)) = 0){
      var objkey;
      objkey = instance_create(x,y,obj_enemy_bullet);
      with (objkey)
motion_set(point_direction(x,y,obj_player_key.x,obj_player_key.y),10);

      }

      action_potential_step(obj_player_key.x,obj_player_key.y,4,0);

}
```

Anexo C-Objecto player

Sprite: spr_player

Visible

Persistant

Step event:

```
mouse_direction = point_direction(x,y,mouse_x,mouse_y);
image_angle = mouse_direction;
if(mouse_check_button_pressed(mb_left)){
    x2 = x+lengthdir_x(10,mouse_direction);
    y2 = y+lengthdir_y(10,mouse_direction);

     var bullet;
    bullet = instance_create(x2,y2,obj_bullet);
    bullet.speed = 10;
    bullet.direction = mouse_direction;
    bullet.image_angle = mouse_direction;
}
//manager dos movimentos
movement();
```

Anexo D-Objecto player_laser

Sprite: spr_player

Visible

Persistant

Step event:

```
mouse_direction = point_direction(x,y,mouse_x,mouse_y);


image_angle = mouse_direction;


if(mouse_check_button_pressed(mb_left)){
    x2 = x+lengthdir_x(10,mouse_direction);
    y2 = y+lengthdir_y(10,mouse_direction);


    var bullet;
    bullet = instance_create(x2,y2,obj_laser);
    bullet.speed = 20;
    bullet.direction = mouse_direction;
    bullet.image_angle = mouse_direction;
}
//manager dos movimentos
movement();
```

Anexo E-Objecto player_key

Sprite: spr_player_key

Visible

Persistant

Step event:

```
mouse_direction = point_direction(x,y,mouse_x,mouse_y);


image_angle = mouse_direction;


if(mouse_check_button_pressed(mb_left)){
    x2 = x+lengthdir_x(10,mouse_direction);
    y2 = y+lengthdir_y(10,mouse_direction);



    var bullet;
    bullet = instance_create(x2,y2,obj_bullet_key);
    bullet.speed = 10;
    bullet.direction = mouse_direction;
    bullet.image_angle = mouse_direction;
}
```

Para todos os objectos player se a vida for inferior a zero executa a linha de código instance_destroy();

Anexo F-Objecto enemy

Sprite: spr_enemy

Visible

Create event:

life = 10;

step event:

detection_player();

detection_laser();

detection_key();

Anexo G -Objecto turret

Sprite: spr_turret

Visible

Create event:

```
turret_life = 200;
```

Step event:

```
if(turret_life <= 0){
    instance_destroy();
    sound_play(rpg);
}
if(distance_to_object(obj_player) <= 400){
       player_direction = point_direction(x,y,obj_player.x,obj_player.y);
        direction = player_direction;
        image_angle = player_direction;
       if(floor (random(20 - 1)) = 0){
        var obj;
        obj = instance_create(x,y,obj_enemy_bullet);
        with (obj) motion_set(point_direction(x,y,obj_player.x,obj_player.y),10);
        }
  }

if(distance_to_object(obj_player_key) <= 400){

       playerb_direction = point_direction(x,y,obj_player_key.x,obj_player_key.y);
       direction = playerb_direction;
       image_angle = playerb_direction;
```

```
    if(floor (random(20 - 1)) = 0){
        var obj;
        obj = instance_create(x,y,obj_enemy_bullet);
        with (obj) motion_set(point_direction(x,y,obj_player_key.x,obj_player_key.y),10);
        }
}
if(distance_to_object(obj_player_laser) <= 400){
        playerb_direction = point_direction(x,y,obj_player_laser.x,obj_player_laser.y);
        direction = playerb_direction;
        image_angle = playerb_direction;
    if(floor (random(20 - 1)) = 0){
        var obj;
        obj = instance_create(x,y,obj_enemy_bullet);
    with (obj) motion_set(point_direction(x,y,obj_player_laser.x,obj_player_laser.y),10);
        }
}
```

Anexo H-Objecto enemy_forest

Sprite: spr_forest_enemy

Visible

Create event:

```
forest_life = 100;
```

Step event:

```
if(forest_life <= 0){
   instance_destroy();
}

if(distance_to_object(obj_player) <= 200){
      player_direction = point_direction(x,y,obj_player.x,obj_player.y);
      direction = player_direction;
      image_angle = player_direction;

   if(floor (random(20 - 1)) = 0){
      var obj;
      obj = instance_create(x,y,obj_forest_bullet2);
      with (obj) motion_set(point_direction(x,y,obj_player.x,obj_player.y),10);

      }
}

if(distance_to_object(obj_player_key) <= 200){

      playerb_direction = point_direction(x,y,obj_player_key.x,obj_player_key.y);
```

```
        direction = playerb_direction;
        image_angle = playerb_direction;

    if(floor (random(20 - 1)) = 0){
        var obj;
        obj = instance_create(x,y,obj_forest_bullet2);
        with (obj) motion_set(point_direction(x,y,obj_player_key.x,obj_player_key.y),10);
        }
}
if(distance_to_object(obj_player_laser) <= 200){
        playerb_direction = point_direction(x,y,obj_player_laser.x,obj_player_laser.y);
        direction = playerb_direction;
        image_angle = playerb_direction;

    if(floor (random(20 - 1)) = 0){
        var obj;
        obj = instance_create(x,y,obj_forest_bullet2);
        with (obj)
motion_set(point_direction(x,y,obj_player_laser.x,obj_player_laser.y),10);
        }
}
```

Anexo I-Objecto small_tank

Sprite: spr_tank

Visible

Create event:

```
tank_life = 300;
```

Step event:

```
if(tank_life <= 0){
   instance_destroy();
   sound_play(rpg);
}

if(distance_to_object(obj_player) <= 400){
      player_direction = point_direction(x,y,obj_player.x,obj_player.y);
      direction = player_direction;
      image_angle = player_direction;

   if(floor (random(20 - 1)) = 0){
      var obj;
      obj = instance_create(x,y,obj_forest_bullet2);
      with (obj) motion_set(point_direction(x,y,obj_player.x,obj_player.y),10);

      }
}
```

```
if(distance_to_object(obj_player_key) <= 400){
        playerb_direction = point_direction(x,y,obj_player_key.x,obj_player_key.y);
        direction = playerb_direction;
        image_angle = playerb_direction;

    if(floor (random(20 - 1)) = 0){
        var obj;
        obj = instance_create(x,y,obj_forest_bullet2);
        with (obj) motion_set(point_direction(x,y,obj_player_key.x,obj_player_key.y),10);
        }
}

if(distance_to_object(obj_player_laser) <= 400){
        playerb_direction = point_direction(x,y,obj_player_laser.x,obj_player_laser.y);
        direction = playerb_direction;
        image_angle = playerb_direction;

    if(floor (random(20 - 1)) = 0){
        var obj;
        obj = instance_create(x,y,obj_forest_bullet2);
        with (obj)
motion_set(point_direction(x,y,obj_player_laser.x,obj_player_laser.y),10);
        }




}
```

Anexo J-Objecto Boss

Sprite: spr_boss

Visible

Create event:

```
boss_life = 6000;
```

Step event:

```
if(boss_life <= 0){
   sound_play(rpg);
   instance_destroy();
   room_goto(room11);
}

if(distance_to_object(obj_player) <= 600){
      player_direction = point_direction(x,y,obj_player.x,obj_player.y);
      direction = player_direction;
      image_angle = player_direction;

   if(floor (random(20 - 1)) = 0){
      var obj;
      obj = instance_create(x,y,obj_enemy_bullet);
      with (obj) motion_set(point_direction(x,y,obj_player.x,obj_player.y),10);

      }

   }
```

```
if(distance_to_object(obj_player_key) <= 400){
        playerb_direction = point_direction(x,y,obj_player_key.x,obj_player_key.y);
        direction = playerb_direction;
        image_angle = playerb_direction;

    if(floor (random(20 - 1)) = 0){
        var obj;
        obj = instance_create(x,y,obj_enemy_bullet);
        with (obj) motion_set(point_direction(x,y,obj_player_key.x,obj_player_key.y),10);
        }
}

if(distance_to_object(obj_player_laser) <= 400){

        playerb_direction = point_direction(x,y,obj_player_laser.x,obj_player_laser.y);
        direction = playerb_direction;
        image_angle = playerb_direction;

    if(floor (random(20 - 1)) = 0){
        var obj;
        obj = instance_create(x,y,obj_enemy_bullet);
        with (obj)
motion_set(point_direction(x,y,obj_player_laser.x,obj_player_laser.y),10);
        }
}
```

Anexo K-Objecto Boss Forest

Sprite: spr_boss_forest

Visible

Create event:

```
boss2_life = 7000;
```

Step event:

```
if(boss2_life <= 0){
    sound_play(rpg);
    instance_destroy();
    room_goto(room13);
}


if(distance_to_object(obj_player) <= 600){
        player_direction = point_direction(x,y,obj_player.x,obj_player.y);
        direction = player_direction;
        image_angle = player_direction;

    if(floor (random(20 - 1)) = 0){
        var obj;
        obj = instance_create(x,y,obj_forest_bullet2);
        with (obj) motion_set(point_direction(x,y,obj_player.x,obj_player.y),10);


        }



    }
```

```
if(distance_to_object(obj_player_key) <= 400){

        playerb_direction = point_direction(x,y,obj_player_key.x,obj_player_key.y);
        direction = playerb_direction;
        image_angle = playerb_direction;

    if(floor (random(20 - 1)) = 0){
        var obj;
        obj = instance_create(x,y,obj_forest_bullet2);
        with (obj) motion_set(point_direction(x,y,obj_player_key.x,obj_player_key.y),10);
        }
}

if(distance_to_object(obj_player_laser) <= 400){
        playerb_direction = point_direction(x,y,obj_player_laser.x,obj_player_laser.y);
        direction = playerb_direction;
        image_angle = playerb_direction;

    if(floor (random(20 - 1)) = 0){
        var obj;
        obj = instance_create(x,y,obj_forest_bullet2);
        with (obj)
motion_set(point_direction(x,y,obj_player_laser.x,obj_player_laser.y),10);
        }


    }
```

Anexo L-Objecto key

Sprite: spr_key

Visible

Colison event obj_player:

```
instance_destroy();
with(obj_player){
    instance_change(obj_player_key,true);
}
```

Colison event obj_player_laser:

```
instance_destroy();
with(obj_player_laser){
    instance_change(obj_player_key,true);
}
```

Anexo M -Objecto end

Sprite: spr_end

Visible

Collison event obj_player:

```
obj_player.x = target_x;

obj_player.y = target_y;

room_goto(target);
```

Collison event obj_player_laser:

```
obj_player_laser.x = target_laser_x;

obj_player_laser.y = target_laser_y;

room_goto(target_laser);
```

Collison event obj_player_key:

```
obj_player_key.x = target_key_x;

obj_player_key.y = target_key_y;

room_goto(target_key);
```

As variaveis são criadas e executadas no jogo.

Anexo N -Interacções

Em relação aos objectos de muralha simplesmente quando ocorre uma colisão executam o seguinte código em certas condições:

Em relação aos inimigos e o player:

move_bounce_all(true);

Em relação aos projécteis:

instance_destroy();

A interacção entre player e inimigos contém a seguinte linha de código para os inimigos:

Instance_destroy();

Para o player:

Executa uma funcionalidade interna da ferramenta que decrementa a sua vida, se a virificação dar como resultado inferior a zero desenhando na barra de vida:

Instance_destroy();

No que diz respeito á balas do player quando entram em contacto com o inimigo executa o código:

instance_destroy();

e decrementa a variável life de cada inimigo criado.

Anexo O –Arma Laser

```
instance_destroy();

with(obj_player){

   instance_change(obj_player_laser,true);

}
```