**ENTREGA FINAL**

ESCOLA SUPERIOR DE TECNOLOGIA IPCA


Pedro Silva nº14281
Tiago Mariano nº13211
Diogo Portela nº13236
Nelson Gonçalves nº10242
Grupo 5 - *Grupo 4

## Introdução

Neste documento falamos sobre o jogo criado para a unidade curricular de Técnicas de Desenvolvimento de Jogos, *Path to Portal*. Serve o presente documento para explicar e exemplificar as ideias, construídas ao longo do semestre, que levaram a criação deste jogo.

## Descrição do Jogo

*Path to Portal* é um jogo *SplitScreen Rogue-like Dungeon Crawler* ou seja, lutamos pela nossa sobrevivência rodeados de monstros em conjunto com mais um jogador, dividindo o ecrã e, para apimentar a situação, cada vez que um jogador morre ambos perdem e veem-se obrigados a recomeçar do início.

A ideia para o jogo surgiu a partir dos jogos que estávamos a jogar no início do semestre, juntamente com a nossa vontade de querer fazer algo que puxasse por nós em termos de codificação.

Este jogo tem como cenário um ambiente semi-medieval, reinado por magia e fantasia, no qual se inserem as nossas personagens principais, dois irmãos gêmeos conhecidos pela sua bravura e força. Devido a acontecimentos estranhos, certo dia, visitaram a floresta azul para confrontar essas ocorrências, o que não sabiam é que ficaram presos para sempre num ciclo.

Não existe ganhar, não há fim e, portanto, não há vitória. Contudo, usamos um sistema de pontuação, cada monstro que um jogador mata torna mais próximo a abertura do portal para o próximo nível, e aumenta o seu contador individual de monstros mortos.

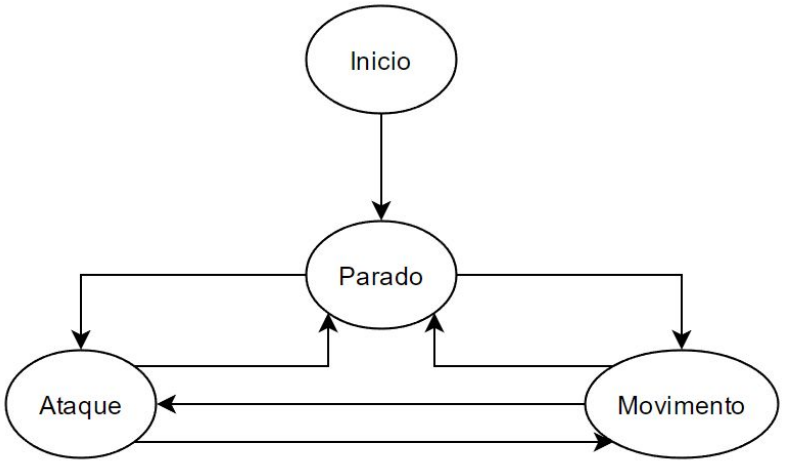
Da nossa lista de jogos de inspiração, e sem querer obviamente descartar todos os outros jogos que escolhemos previamente, pensamos que o mais próximo poderá ser *Tiny Wizard*. Escolhemos este pois *Tiny Wizard* contém uma sensação de espacialidade semelhante, com os seus mapas gerados, movimentos e habilidades de combate.

O nosso objetivo principal é fazer um jogo para ser jogado em máquinas *Arcade*, mas também correr em *Windows* sendo, portanto, o nosso público alvo principal, pessoas que nasceram ou viveram o início dos anos 90, dado o efeito nostalgia dos centros árcade dos anos 80.

Pedro Silva nº14281 Diogo Portela nº13236
Tiago Mariano nº13211 Nelson Gonçalves nº10242
Grupo 5 - *Grupo 4

Mecânica do jogo

Máquina de Estado do jogador:




Nota: No momento da entrega não existem animações para os inimigos, como tal só há máquinas de estado para os jogadores, que são exatamente iguais.

Blocos dos

momento existe um opções, nos nossos futuros.

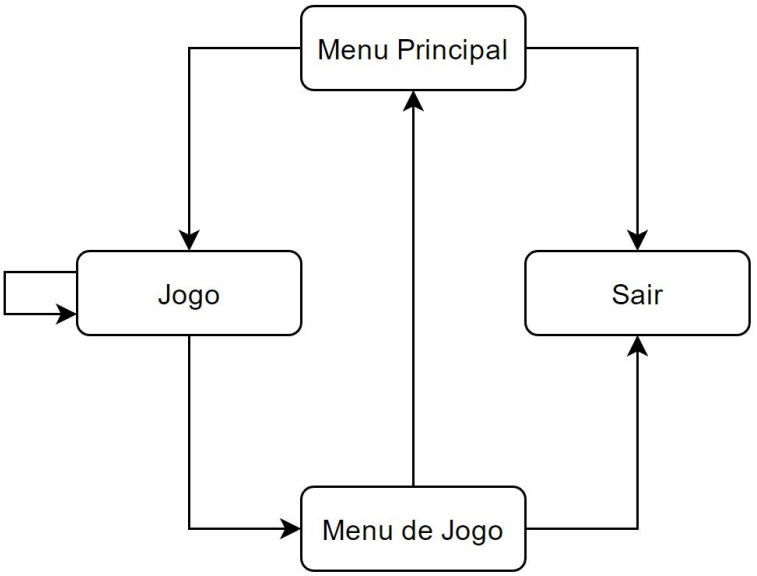



Diagrama de Menus:

Nota: Neste ainda não menu de mas consta objectivos

comandos

Para os do jogo,

inspiramos-nos na disposição das máquinas arcade. Usamos botões próximos, mais um conjunto de botões direcionais para obter o efeito.

Como tal, WASD e as setas servem de movimento para o jogador um e dois, correspondente aos *joysticks* de uma arcade. Espaço VBN servem de botões extra para o jogador um (sendo que, ao nível actual de desenvolvimento, Espaço é o ataque e V troca de armas) enquanto *Numpad* 0123 são os do jogador dois (da mesma forma, 0 serve para ataque e 1 para trocar de armas). Por fim existem os botões genéricos como o Esc e o F2 que servem

para abrir o menu e desligar o modo *debug*.

Seria também do nosso interesse conseguir aplicar estes inputs aos controles de uma máquina arcade a sério.

## Implementação

Neste trabalho, utilizamos a combinação de C#, uma linguagem extremamente versátil (dentro das linguagens de alto nível) com a plataforma XNA - *Monogame*.

Desde o primeiro momento em que discutimos o que viria a ser o nosso jogo, que pensámos desenvolvê-lo por forma a que tivesse vários sistemas únicos que permitissem facilmente e a qualquer momento, adicionar algo mais, por forma a podermos enriquecer o jogo. Neste sentido, criámos algo que se assemelha mais a um protótipo, que nos permitiu explorar as potencialidades do *Monogame,* a ponto de conseguirmos desenvolver tudo pela raíz. Para chegarmos a esse ponto, para além de aprofundarmos alguns conceitos pela elaboração do primeiro trabalho da cadeira, tivemos de aprender a manipular a nosso favor a simples, mas extremamente eficiente *framework* XNA. Para além disso, visto que o *Monogame* tem uma fraca documentação, optámos por fazer uso dos tutoriais do *Unity* visto que é um *game engine* que, para além de também ter como base o C#, desfruta de uma comunidade bastante envolvida e dedicada em publicar pequenos vídeos, que explicam alguns conceitos importantes e que estão por detrás do que seguidamente vem descrito.

Ora, sendo de um protótipo que estamos a falar, parece-nos mais adequado falar dos sistemas que criamos nesta secção do relatório e não tanto de funções que interagem com o utilizador. Neste sentido, abordaremos levemente os vários sistemas criados por nós por forma a tornar a leitura simples e minimamente agradável. Para leitores mais interessados em aprofundar estes conceitos ou que simplesmente queiram olhar para o código produzido, deixaremos sempre, no nome das classes, links para o *GitHub*.

Começamos pela *Camera*. Esta simples *class* permite-nos situar o jogador numa determinada posição do que já está desenhado no ecrã. Especificamente para este jogo, permite-nos criar uma visão *splitscreen* em que cada jogador tem uma instância da *class* câmera que o segue.

Seguidamente olhamos para a *class Tile* que é uma herança da classe *GameObject,* o que lhe atribui automaticamente uma textura, posição e tamanho. Desta forma,

atribuímos-lhe, para além destes parâmetros, uma coordenada e um número. Consoante o número atribuído, para este jogo em específico, ela irá receber uma cor, de forma aleatória, entre o azul e o violeta. Esta classe será utilizada como base das duas próximas classes.

Posteriormente, foi criada uma classe que gera aleatoriamente o mapa - *MapGenerator* que, em conjunto com a *class Tile* e uma *class* auxiliar *Room* formam o sistema de *tiles*. Para criar o mapa, recorremos ao modelo da matemática finita - *Cellular Automata*. De forma a contextualizar o leitor, este modelo actualiza o estado de cada célula através de algumas regras determinísticas, com base no estado actual das células envolventes. Na prática, isto permite-nos, após encher a nossa matriz de *tiles* com 0's e 1's (sendo 0 um espaço vazio e 1 uma parede) e correndo as regras definidas por nós, que cada *tile* se actualize até termos um mapa automaticamente e aleatoriamente concluído.

*Procedual map generation*

*Final iteration*

Para conseguirmos que os jogadores andem apenas no espaço vazio e não em cima de paredes, tivemos de criar um sistema de colisões com base nas *tiles*. Daí surge a *class Collider*. Esta *class* toma como exemplo o modelo usado no *MapGenerator* para definir atualizações na *tile* em que cada jogador se posiciona em cada iteração do *gameloop*. Por exemplo, se um jogador está próximo de uma parede e usa as teclas de movimento para se deslocar para a mesma, este irá colidir e parar. Isto acontece porque todas as *tiles* estão em constante actualização e quando correrrmos a função *UpdateDelta*, que recebe o vector de direcção do jogador, este irá verificar que na posição actual do jogador + direcção estará uma *tile* no qual o jogador não pode andar, bloqueando completamente o *input* naquele sentido.

Após completarmos as *core mechanics* do jogo, passamos a criar sistemas para embelezar o jogo em si. Em vez de usarmos xml para animações ou simplesmente corre-las em *loop* no meio do código, o que o iria tornar difícil de ler e pouco organizado, optámos por criar o nosso próprio sistema de animações.

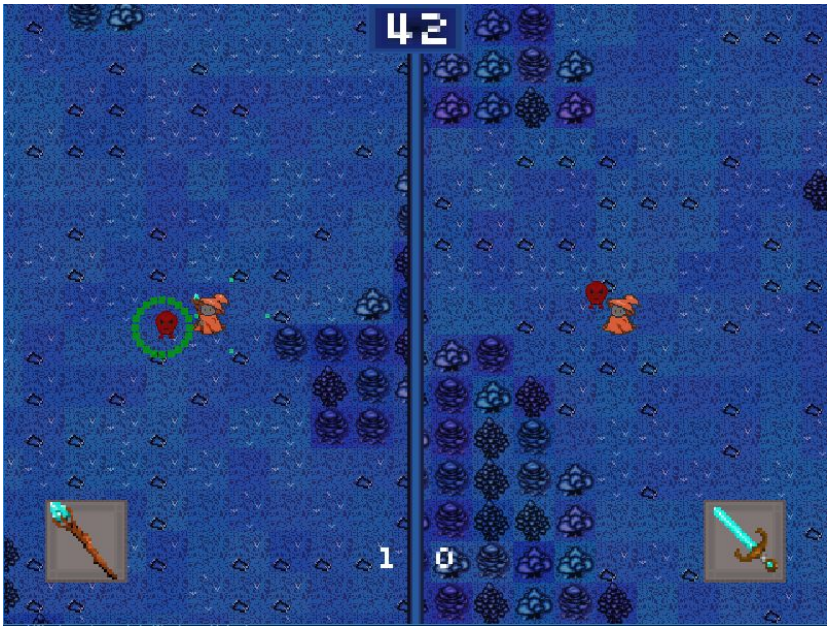
*Animations* permite ao programador, com auxílio de uma instância por ele criada,

percorrer uma *spritesheet* com a simples atribuição da função *.Play*. Quando quiser parar a animação (seja porque quer interrompê-la a meio ou simplesmente porque a animação já acabou) basta correr *.Stop*. Isto torna o trabalho com animações completamente trivial.

Para além das animações, e como queríamos atribuir alguns ataques com alcance aos jogadores, criámos um sistema de partículas. Este sistema, não só nos permite poupar trabalho em animações que agora podem ser feitas com pequenas partículas, como as podemos usar para colidirem com inimigos (cada partícula é vista como um *GameObject*) e dar dano no impacto. *ParticleSystem* é o responsável por estas acções.

Para interação visual com o utilizador, criámos também o *UI* que mostra não só o *score* de cada jogador, como a sua arma atual e quantos monstros faltam para acabar o nível.

Para concluir e por conveniência, criámos uma *class Debug* que imprime para o ecrã quaisquer informações que o programador ache convenientes no actual progresso de desenvolvimento.




Como nota final, vale a pena referir que começamos por usar *classes* de *pathfinding opensource* disponíveis no *GitHub* mas por peso excessivo para uma *arcade*, acabamos por criar o nosso próprio *Pathfinder* com recurso ao que aprendemos sobre a teoria dos grafos na cadeira de Paradigmas da Programação II.

Arte

Toda a arte usada no jogo é original assim como o som, que foi feito por nós. Achamos que o estilo mais devido seria *pixel art*, quer pela sua baixa intensidade de processamento no computador como pela sua aproximação ao estilo de jogos das máquinas *Arcade*.

Numa fase inicial consideramos entre criar o nosso jogo numa perspectiva isométrica, pesamos as dificuldades que achamos que iríamos ter com a construção deste jogo e achamos que seria melhor apontar para uma perspectiva frontal, pois esta não nos causaria tantos problemas em gerar mapas e até mesmo na construção de assets.

Consideramos também usar um processo de colorização de cada instância de uma dada textura para tornar o jogo mais atrativo ao mesmo tempo que reduzimos o custo de produção de *assets*. Este processo consiste principalmente em criar *assets* todos numa cor neutra e idêntica de modo a que mais tarde, para criar diferenciação, bastava aplicar um filtro de cor por cima alterando assim a sua cor final, por exemplo, caso o jogador tivesse uma arma de fogo o seu sprite tornaria-se mais vermelho. No entanto, a implementação deste processo ficou pela texturas de background de modo a criar vivacidade e eliminar a sensação de repetição.

Imagens para o fundo.

A personagem foi sendo repensada ao longo do jogo. O seu primeiro sprite foi feito duma maneira crua e imediata para servir de *placeholder*, sendo mais tarde substituída pelo seu primeiro teste (não presente pois a imagem foi apagada). O problema desta versão foi o facto de ter sido feita numa escala diferente, logo ficou com problemas de resolução e a sua cor também não era a melhor pois sendo azul confundia-se com o fundo facilmente. Por fim resultou a última versão, mais arredondada, pequena e com uma cor contrastante com o fundo.

Personagens.

As armas também tiveram versões anteriores, mas estas serviram unicamente de *placeholders* sem qualquer relação com o produto final. No fim optamos por armas que tivessem alguma sensação de mística e mágia.

Armas usadas.

Desenvolvimento

A nossa ideia inicial não foi completamente alterada durante o processo, no entanto, como já foi descrito ao longo do relatório, foi sofrendo alterações. Desde a nossa expectativa do aspecto gráfico que foi mudando ao longo do percurso, como por exemplo a nossa vontade de querer ter animações diferentes dependendo de cada arma que o utilizador tem, cada jogador ter uma cor diferente, termos biomas diversificados e monstros diferentes para cada bioma, até a aspectos mais técnicos como ter um sistema de inventário.

A nossa dificuldade principal que levou ao atraso no desenvolvimento foi coordenar o trabalho, a disparidade das capacidades dos membros do grupo e o respeito pelas datas de entrega.

Segue em lista as datas de entregas e as prestações dos membros do grupo:

31 de Março.
- Mecânicas de movimento, mecânicas de animação, mecânicas de câmera. (Diogo)
- Mecânicas de combate e defesa. (Tiago) Entregue atrasado.
- Mecânicas para gerar mapas. (Pedro)
- Sistema de AI. (Nelson) (Mais tarde reiterado pelo Pedro)
- Colisões (Pedro e Diogo)

30 de Abril.
- Sistema de Partículas. (Diogo)
- Sistema de Som. (Tiago)
- Sistema de gerar monstros (Pedro)
- Mecânicas de troca de armas. (Nelson)

Fim de Maio
- Menus (Diogo)
- Efeitos de Combat (Diogo e Pedro)

Não nos foi possível cumprir a semana de teste.

Trailer (Diogo)
Relatório (Diogo e Pedro)

## Desenvolvimento Futuro

No nosso jogo ainda sobraram algumas ideias fundamentais para implementar, um sistema de inventário para poder trocar de objectos, armas, poções, itens entre jogadores. As referidas poções, objetos e itens, que serviram para modificar o jogo, dando atributos aos jogadores. Gostávamos também de introduzir novas armas que modificasse a jogabilidade, e novos monstros com formas unicas de lutarem.

## Conclusão

No fim podemos dizer com certeza que este trabalho foi uma boa experiência para aprofundar os nossos conhecimentos da linguagem C#, assim como, aprender como um jogo corre por trás. Foi interessante ver como se aplicam mecânicas de jogo em código e como podemos, e devemos, usar todos as ferramentas ao nosso dispor para poder "iludir" o jogador.

Um videojogo não passa de uma colecção de ilusões muito bem feitas, em que todas juntas trazem magia ao ecrã dos nossos computadores e nos fazem envolver numa experiência psicológica e emocional.