# Wave-Born

Pedro Iglésias 89318, Wei Ye 93442

## Funcionamento do jogo

Pode-se controlar o player atrávez das teclas 'a', 'd', e 'space', as primeiras duas movem o player para esquerda ou direita enquanto a última é para saltar.

O mundo está totalmente escuro, não se consegue ver nada, nem paredes nem monstros. Quando um monstro começar a atacar o player ou simplemente gritar, uma onda que revela um pedaço do mundo é gerada. Além disso, o movimento do player também gera ondas.

Qualquer monstro pode matar o player e vice-versa.

## Arquitetura

chunks: contém ficheiros json para geração do mundo.

menu: contém classes de menu principal, configuração e pause.

models: contém objetos usados no jogo como player, monster e wave, etc.

sources: contém imagens e sons que o jogo precisa.

sprites: contém classes de sprite.

# Padrões

## Command

- Command foi usado para interpertar os comandos do utilizador a partir das teclas configuradas a partir do menu.

A aplicação do padrão encontra-se no ficheiro *models/player.py entre as linhas 13 e 15*. As classes de comandos estão no ficheiro *models/common.py a partir da linha 23 até a linha 63*.

## Flyweight

- Flyweight foi usado em quase todos sprites tanto na geração do mundo como nos monstros, para reutilizar as texturas em vez estar sempre a cria-las.

Encontra-se a aplicação em quase *todos sprite files*.

## Prototype

- Prototype foi usado para monstros e diferenciar entre monstros voadores e de chão. Como também um spawner para fazer spawn aos monstros.

Encontra-se principalmente no ficheiro *models/monsters.py*.

## Singleton

- Todos os sprites são singletons pois não faz sentido ser criado varios objetos sprites quando a textura é sempre a mesma que esta a ser usada e a única diferença é que temos um novo monstro modelo a ser adicionado ou removido desta classe. O world é um singleton pois o mundo é sempre o mesmo e os chunks que são usados para gerar o mundo são sempre os mesmos a única diferença é a ordem dos chunks gerados.

Encontra-se principalmente nos ficheiros *models/world.py* e *sprites/monsters_sprites.py*.

## State

- States foram usados para saber todas transições e estados em que o player e os monstros estão.

- FSM foi usado para controlar transições entre estados.

-Monster State:

Um monstro em geral tem 3 estados: Move, Attack e Dead.

No estado Move, o monstro move-se horizontalmente.

Existe uma probabilidade x de passar do estado Move para Attack.

O montro pode morrer em qualquer estado.

A aplicação encontra-se no *models/monsters.py* o método *update* de todos monstros voadores, nomeadamente *BirdLike (linha 215)* e *WhaleLike (linha 432)*.

-GroundMonster State:

Um monstro em geral tem 6 estados: Move, Attack, Jump, Fall, MoveInAir e Dead.

No estado Move, o monstro move-se horizontalmente.

Existe uma probabilidade x de passar do estado Move para Attack.

O monstro pode morrer em qualquer estado.

MoveInAir acontece quando o monstro sai do bloco, ou seja, está a caminhar no ar.

A aplicação do padrão encontra-se no *models/monsters.py* o método *update* de todos monstros de chão, nomeadamente *SpriderLike (linha 271)* e *TurtleLike (linha 271)*.

### Game loop

- Game loop foi usado para cada criação de um mundo.

- Passando por "process input": onde os eventos e o input do teclado do player é processado.

- Depois por "update game": onde tudo no jogo é updated os monstros, o player, o mundo e a camera.

- Por fim passa por "render": onde os sprites todos são renderizados.

A aplicação do padrão encontra-se no ficheiro *main.py a partir da linha 34*.

# Aspectos “Inovadores”

Os aspetos mais interessantes do jogo são as ondas de som que revelam parte do mapa e a geração do mundo.

O som revelar o mapa de um jogo é usado em diversos jogos mas nunca tinhamos visto num platformer por isso é que decidimos aplicar neste jogo.

A geração do mundo é feito antes de começar um new game e utiliza ficheiros existentes de chunks pré-fabricados e monta um mapa possivel de completar do inicio ao fim a partir de condições no ficheiro pré-determinadas.

Depois o mundo como já tem os chunks escolhidos e gerados só precisa de carregar os 3 chunks a volta do player (o chunk em que o player esta 2 para tras e 2 para a frente) funcionando como uma conveyor belt onde o player caminha numa direção e vai carregando novos chunks à medida que caminha.

# Github Link

O repositório para o projeto: https://github.com/Iglesias-Leafwind/Wave-Born/tree/master

# Referências

Inspiração da mecânica do som, o jogo “Dark Echo”:

- https://www.youtube.com/watch?v=tuOC8oTrFbM

Sprites:

- Background.png -> https://wallpapersden.com/cyberpunk-city-pixel-art-wallpaper/1360x768/
- Bird.png -> https://www.nicepng.com/ourpic/u2q8q8i1q8w7r5i1_sprite-sheet-bird-png/
- End_city.png -> https://www.deviantart.com/mysticmorning/art/Sci-Fi-Fantasy-Building-2-359889492
- Feather.png -> http://clipart-library.com/clip-art/transparent-feather-21.htm
- Player.png -> https://www.pixilart.com/art/2d-player-sprite-sheet-317ef5787732657
- Spider.png -> https://www.pngitem.com/middle/hixJxbT_spider-0-spider-sprite-animation-sheet-hd-png/
- Tortoise.png -> https://www.spriters-resource.com/ds_dsi/finalfantasy12revenantwings/sheet/424/
- Whale.png -> https://opengameart.org/content/swimming-whale

Sons:

- Bird.mp3 -> https://pixabay.com/sound-effects/gryffin-cry-6995/
- Breeze_bay.mp3 -> https://soundcloud.com/hellometeor/breeze-bay
- Jump.mp3 -> https://pixabay.com/sound-effects/swing-whoosh-110410/
- Land.mp3 -> https://pixabay.com/sound-effects/land2-43790/
- Running.mp3 -> https://pixabay.com/sound-effects/running-1-6846/
- Step.mp3 -> https://pixabay.com/sound-effects/footsteps-grass-1-6810/
- Turtle.mp3 -> https://pixabay.com/sound-effects/sleeping-monster-38084/
- Whale.mp3 -> https://pixabay.com/sound-effects/long-howl-whale-and-monster-37270/