# Dung!

## Documento de Design do Jogo

## Resumo

Dung! é um jogo do tipo Roguelite devido a vários aspetos característicos deste género, como a escolha de classes de personagens, as recompensas recebidas ao longo do jogo, as variedades de itens existentes e a morte permanente. O Dung! junta o conceito de Roguelites com o antigo e icónico jogo dos anos 70, o Pong, onde o objetivo era refletir uma bola quase como um jogo de ténis e esperar que o adversário não conseguisse fazer o mesmo.

No nosso jogo o jogador pode então escolher uma de 3 classes que influenciam o gameplay. Em termos de combate, o jogador tem de refletir os projéteis disparados pelo inimigo no tempo certo de modo a que o inimigo perca vida. Ao longo do jogo o jogador é recompensado com vários itens que facilitam o seu progresso.

## Classes de Personagens

Como mencionado existem 3 classes no jogo, estas são, o guerreiro, o arqueiro e o mago. Estas classes diferenciam-se na quantidade de vida e ataque que têm.

| | **Guerreiro** | **Arqueiro** | **Mago** |
|---|---|---|---|
| **Vida** | 6 | 4 | 2 |
| **Ataque** | 2 | 3 | 4 |

Ao refletir um projétil o jogador gasta estamina, que é naturalmente recuperada ao longo de meio segundo. O mago é uma pequena exceção a este tempo como vamos ver a seguir.

Para além disto, cada classe também possui um ataque especial. Da mesma forma que para refletir projéteis é preciso estamina, estes ataques só podem ser usados quando o jogador tem 6 pontos especiais, ganhos ao refletir projéteis com sucesso.

- O guerreiro gera um escudo que reflete automaticamente um projétil caso o jogador não o consiga fazer.
- O arqueiro dispara uma flecha que pode arrastar no seu caminho os projéteis, embora dê mais dano ao inimigo se o acertar diretamente.
- O mago dispara um feitiço que ao acertar no inimigo dá dano e recupera um ponto de vida que tenha sido perdido. Caso o ataque tenha acertado num projétil o mago fica cansado, não conseguindo refletir ataques durante 2 segundos.

## Itens

Existem vários itens no jogo: armas, gemas e poções de vida.

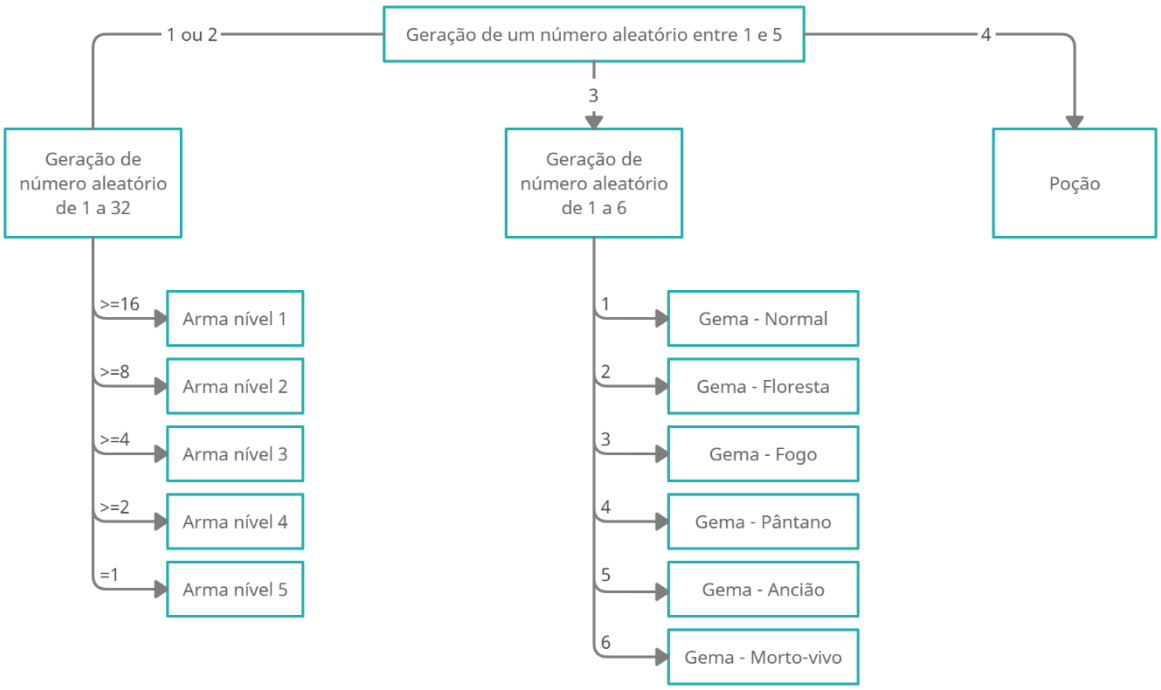
Visualmente, as armas dependem da classe do herói selecionado, por exemplo as armas do mago são bastões enquanto que as do guerreiro são espadas. Em termos de estatísticas, estas dependem do

nível da arma. As armas podem ir desde nível 1 ao nível 5. Uma arma de nível 1 dá +1 de ataque ao jogador sendo que esse valor aumenta por 1 por cada nível.

Em relação às gemas, estas são tantas quanto os tipos de inimigos, que são, normal, floresta, fogo, pântano, ancião e morto-vivo. As gemas permitem dar o dobro do dano aos inimigos do tipo correspondente.

A poção de vida possibilita que, durante o combate, o jogador recupere 1 dos pontos de vida perdidos.

Todos estes itens podem ser obtidos ao derrotar um boss. O algoritmo que retorna uma recompensa segue a seguinte ordem de raciocínio:




Através do diagrama é possível concluir que o jogador pode também não receber nada caso o primeiro número gerado seja 5.

## Progressão

No jogo existem rondas que são compostas por níveis. Cada nível é um combate contra um inimigo sendo cada ronda o conjunto de níveis até ao nível do boss (inclusive). Cada vez que um inimigo é derrotado, naturalmente o nível aumenta, sendo que o número de níveis é tanto quanto o valor da ronda + 2 não incluindo o nível do boss.

A quantidade de vida dos inimigos básicos varia segundo a equação:

$$vidaInimigo = 2 * ronda + nível$$

Sendo que a esse valor é adicionado 6 caso o inimigo seja um boss.

Inicialmente a velocidade dos projéteis é de 1 pixel por frame. Este valor é incrementado por 0.2 a cada ronda concluída parando nos 3 píxeis.

Mas se os píxeis são números inteiros o que quer isto dizer visualmente? As posições decimais são arredondadas para números inteiros, mais concretamente arredondadas para baixo (uma posição 3.4 corresponde ao pixel 3). Isto só significa que alguns píxeis são ignorados, por exemplo:

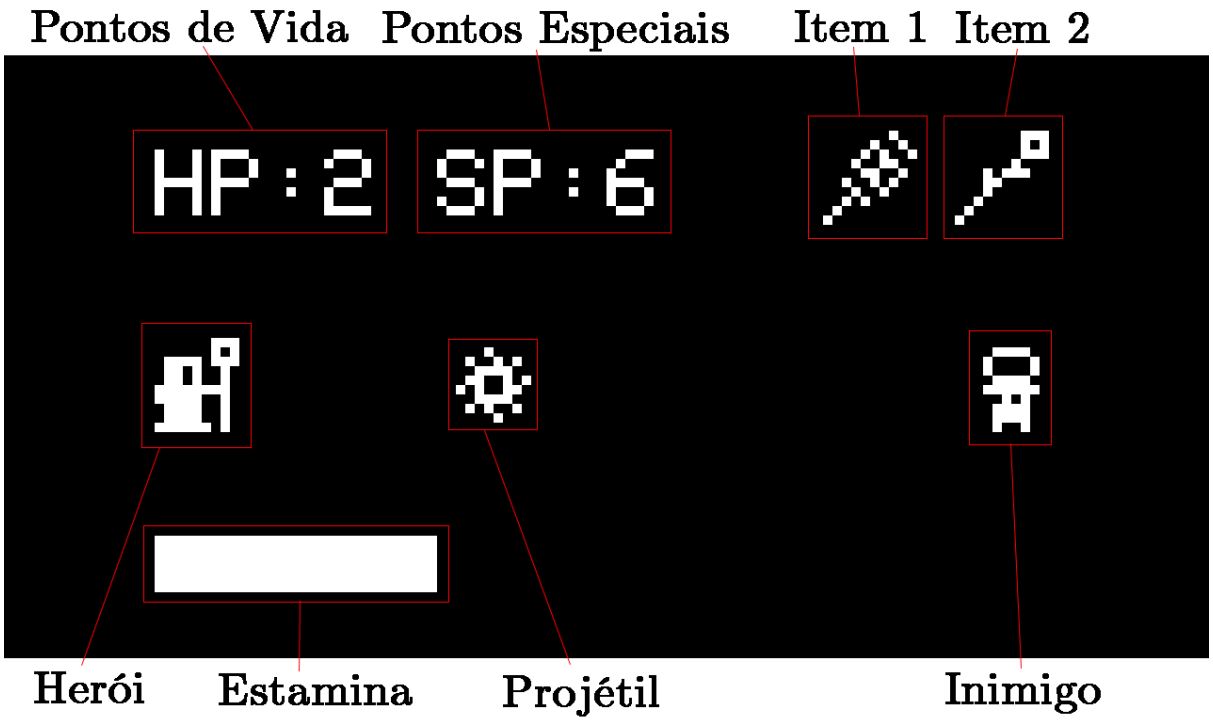
Se a velocidade for 1.2 píxeis por frame, um projétil que se mova do pixel 0 ao 6 vai ignorar o pixel 5, de acordo com os arredondamentos.

| **Valor calculado** | 0 | 1.2 | 2.4 | 3.6 | 4.8 | 6 |
|---|---|---|---|---|---|---|
| **Pixel** | 0 | 1 | 2 | 3 | 4 | 6 |

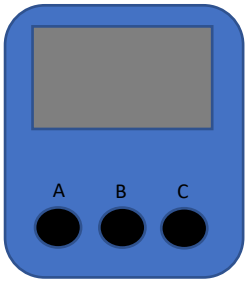
## High scores

No Dung!, a consola guarda na sua EEPROM as três melhores pontuações. Quando o jogador perde é verificado se a pontuação conquistada é melhor que uma das três registadas. Em caso positivo, a nova pontuação é inserida na posição correspondente.

## Interface



# Controlos




Tendo a consola 3 botões estes vão ter as seguintes funcionalidades:

| | A | B | C |
|---|---|---|---|
| Menus | Mover para a esquerda | Selecionar/Confirmar | Mover para a direita/Voltar |
| Jogo | Refletir projéteis | Utilizar poção | Usar ataque especial |

# Sons e Música

Todos os sons feitos pela consola utilizam um ou vários pares frequência-tempo que correspondem à frequência do som tocado e o tempo que este deve durar. Estas frequências podem também ser representadas por notas para um processo de criação mais fácil.

Quando o jogo é iniciado é tocada a música principal que é composta pelos seguintes pares de notas e tempos.

| **Nota** | **Tempo (ms)** |
|---|---|
| C5 | 205 |
| PAUSA | 205 |
| C5 | 205 |
| G5 | 205 |
| F5 | 205 |
| PAUSA | 205 |
| E5 | 205 |
| PAUSA | 205 |
| **Sequência anterior novamente** | |
| C5 | 205 |
| PAUSA | 1027 |
| E5 | 205 |
| PAUSA | 205 |
| C5 | 205 |
| PAUSA | 1027 |
| E5 | 205 |
| PAUSA | 205 |

Para além da música existem também vários efeitos sonoros que tocam ao longo do jogo.

| Ação | Frequência (Hz) | Tempo (ms) |
|---|---|---|
| Mover para esquerda/direita em menus | 4000 | 100 |
| Confirmar personagem /Recomeçar jogo | 4000 | 100 |
| | 4500 | 100 |
| | 5000 | 100 |
| Projétil gerado | 500 | 100 |
| Projétil refletido | 800 | 200 |
| Inimigo atingido | 600 | 200 |
| Jogador atingido | 100 | 100 |
| Poção usada | 1000 | 200 |
| Especial usado | 4000 | 50 |
| | 2000 | 50 |
| | 1000 | 50 |
| Jogador perdeu | 300 | 200 |
| | 200 | 200 |
| | 100 | 200 |
| Ronda terminada | 3000 | 200 |
| | 3500 | 200 |
| | 4000 | 200 |

# Desenvolvimento

Tendo nós optado por usar o microcontrolador Arduino Nano, naturalmente o desenvolvimento de qualquer software seria feito no Arduino IDE, que usa a linguagem de programação C++.

O código desenvolvido para o Arduino, embora tenha outros usos como o nosso caso, é intendido principalmente para desenvolver interações entre vários componentes hardware que se interligam a partir do microcontrolador escolhido. Por isso para desenvolver um jogo seria extremamente complicado e desgastante usar apenas os recursos nativos do IDE. Decidimos então procurar uam biblioteca que pudesse facilitar o nosso processo de desenvolvimento.

A biblioteca que acabámos por utilizar foi a do Arduboy. Esta biblioteca tem os recursos necessários que permitem desenvolver todos os aspetos que um jogo do estilo retro deve ter.

Todos os gráficos do jogo são representados por sprites, gráficos de computador que representam algo e que podem ser movimentados no ecrã onde são visualizados. No Arduboy estes são representados por bitmaps, arrays de bytes que representam um sprite a ser desenhado no ecrã. Como? Cada byte corresponde a uma coluna vertical de píxeis onde o bit menos significativo é o bit no topo da coluna. Os bits a 1 são colocados com a cor desejada (preto ou branco) e os a 0 não são alterados. Observemos um exemplo:

O array: { B01111110, B10000001, B10010101, B10100001, B10100001, B10010101, B10000001, B01111110 } resulta na seguinte imagem se seguirmos a lógica explicada anteriormente:

Nas funções que desenham estes bitmaps é preciso também especificar a largura e altura da imagem para que se saiba se é preciso descartar bits ou quando é preciso mudar de linha. Vejamos outro exemplo onde as dimensões passadas às funções é 10x3:

{0x05, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x07, 0x02}

Aqui, como é mencionada uma altura de 3 píxeis, os bits mais significativos são ignorados.

Tendo explicado estes aspetos passamos agora às funções desta biblioteca usadas por nós:

- `begin()`:
  Chamada no início do código. Inicializa os pinos dependendo do hardware utilizado, reinicia e limpa o ecrã.
- `initRandomSeed()`:
  Usa a função `randomSeed()` do Arduino para inicializar o gerador de números pseudoaleatórios passando-lhe o valor lido de um pino não conectado.
- `nextFrame()`:
  Indica se está na hora de mostrar a próxima frame desenhada. Esta função depende da frame rate definida (60 por padrão).
- `clear()`:
  Limpa o buffer do ecrã.
- `pollButtons()`:
  Lê e guarda o estado dos botões. As funções `justPressed()` e `justReleased()` verificam se o estado dos botões foram alterados entre chamadas consecutivas desta função de forma a saber se acabaram de ser carregados ou libertados. Para o bom funcionamento do programa esta função deve ser chamada a cada frame.
- `drawBitmap(int16_t x, int16_t y, const uint8_t* bitmap, uint8_t w, uint8_t h, uint8_t color = WHITE)`:
  Desenha o `bitmap` na posição `x,y` com a largura `w`, altura `h` e cor (`color`) como foi explicado anteriormente.
- `print(String text)`:
  Desenha os caracteres da string recebida na posição do cursor.
- `setCursor(int16_t x, int16_t y)`:
  Move o cursor para a posição indicada.
- `display()`:
  Mostra no ecrã o conteúdo do buffer que contem os píxeis desejados.
- `idle()`:
  Põe o CPU em modo de hibernação. Esta função deve ser chamada o mais frequentemente possível de modo a economizar a bateria. No nosso caso é chamada entre renderizações de frames.

Para tocar os sons do jogo são usadas as funções `tone()` e `tones()` da biblioteca ArduboyTones, onde a primeira recebe até 3 frequências a serem tocadas e os seus tempos e a última recebe um array em memória de programa contendo esses pares.

## Ecrãs

Ao longo do jogo são apresentados vários ecrãs com diferentes propósitos:

| Nome | Descrição |
|---|---|
| **Start** | É mostrado o logotipo do jogo. Onde se pode ativar ou desativar o áudio e mostrar as melhores pontuações. |
| **Choose** | Onde o jogador escolhe a classe desejada para jogar |
| **Game** | Aqui ocorrem os eventos do jogo em si |
| **Over** | Ecrã de fim de jogo. É possível recomeçar o jogo ou voltar ao ecrã inicial. |
| **Won** | Mostrado ao derrotar um inimigo com sucesso. |
| **Round Over** | Depois do jogador acabar uma ronda, derrotando um boss, é apresentado este ecrã. |
| **Equip Item** | Exibido logo a seguir ao ecrã de fim de ronda. O jogador escolhe onde equipar o item encontrado. |
| **Highscore** | Ao ser derrotado, o jogador vê este ecrã caso tenha feito uma das melhores pontuações registadas pela consola. |