



Instituto Tecnológico de Aeronáutica
Departamento de Software e Sistemas de Informação (IEC-I)

Documentação do jogo *Nick's Hardest Adventure*

Bryan Diniz Borck
Rodrigo Macêdo Rios
Júlio César Cardoso Duarte

Professores : Karla Donato Fook e Edgar Yano

29/04/2022

1 Descrição

Nick's Hardest Adventure é um jogo de plataforma que tem por objetivo desafiar o jogador a pensar maneiras inteligentes e efetivas de terminar os 3 níveis propostos, que apresentam dificuldades e desafio em ordem crescente, completando o objetivo obrigatório, que é a coleta das 5 estrelas distribuídas ao longo das fases, sem as quais a personagem não pode prosseguir.

O jogo conta com mecânicas clássicas de jogos de plataforma, que giram em torno da movimentação lateral da personagem ao longo do nível, além da funcionalidade de pulo, que permite o alcance das plataformas presentes em cada nível e coleta dos itens disponíveis, além de ser o mecanismo por meio do qual a personagem pode derrotar os inimigos distribuídos por meio da colisão superior.

O projeto teve como principal referência o jogo clássico *Super Mario World*, que serviu de inspiração tanto estética, como de jogabilidade para a implementação das funcionalidades.



Figura 1: Jogo *Super Mario World*.

2 Temática

Nick's Hardest Adventure conta com temática lúdica que gira em torno do desafio que a personagem principal, Nick, tem em enfrentar diferentes cenários, divididos em deserto, floresta e ambiente nevado e coletar as estrelas distribuídas em cada um deles, além dos diamantes presentes, derrotando os inimigos e prosseguindo em sua jornada.

Além disso, cabe ressaltar que a temática gira em torno da progressão de dificuldade que a personagem principal encontra com o decorrer dos cenários propostos, de

modo que, de maneira inteligente, ela precise concluir os níveis de forma estratégica os níveis sem perder o número de vidas e corações disponíveis, de modo a não ter de retornar sua jornada do início.



Figura 2: Temática do jogo.

3 Jogabilidade

3.1 Menu e Mecânicas

Nick's Hardest Adventure conta com um menu inicial em que é possível selecionar as opções Jogar, Opções e Sair.

O modo Opções redireciona o jogador a uma tela em que é possível a seleção do modos mutado e não mutado, além de uma funcionalidade extra, não implementada devido as prioridades dadas a outras funcionalidades e restrições de tempo para a finalização do projeto, que poderia vir a ser efetivada, relativa a possibilidade da escolha de um personagem principal masculino ou feminino.

O modo Sair fecha a janela relativa ao jogo e o encerra.

O modo Jogar redireciona o jogador para uma tela de explicação relativa às mecânicas presentes no jogo. As mecânicas presentes são:

- → : movimenta o personagem para a direita.
- ← : movimenta o personagem para a esquerda.
- Space: provoca o movimento de salto da personagem.
- Esc: pausa o jogo.

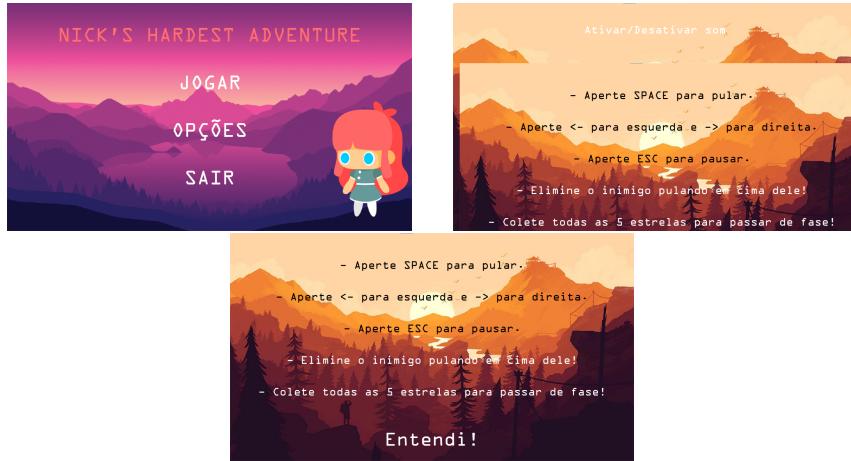


Figura 3: Menu do jogo.

Destaque: A mecânica de derrota dos inimigos distribuídos ao longo dos níveis foi implementada de modo que fosse provocada pela colisão superior da personagem com o mesmo, ou seja, de modo que ao saltar e pousar sobre a cabeça do inimigo, sua destruição fosse efetivada.

Além disso, a colisão lateral da personagem com o inimigo provoca um dano, refletido na diminuição do número de Heart disponíveis, explicado logo em seguida. Em adição, ao cair em um buraco disponível em um nível, ocorre morte automática e o personagem é redirecionado ao inicio do nível.

3.2 Power-Ups e Power-Downs

Nick's Hardest Adventure conta com uma série de Power-Ups e Power-Downs que facilitam e dificultam a personagem principal de cumprir seu objetivo de atravessar os cenários disponíveis.

Tais elementos estão divididos em:

- Diamond : coletáveis não obrigatórios que alteram o *score* obtido pelo jogador no jogo. Caso ocorra a morte da personagem, o valor de *score* retorna a 0.
- Star : coletáveis obrigatórios distribuídos ao longo dos níveis essenciais para a conclusão da fase. Ao todo, por nível são 5 disponíveis para a coleta.
- Heart: coletável que aumenta o número de corações possuídos pela personagem. Limitado em um valor máximo de 5. Caso seu valor seja reduzido a 0, ocorre a morte automática da personagem e ela retorna ao inicio do nível.

- Lives: coletável que altera o número de vidas disponíveis pela personagem para completar todos os níveis. Caso seu valor seja reduzido a 0, a morte automática da personagem e a jornada é reiniciada, ou seja, há o retorno ao nível 1.
- Invencibility: coletável que permite com que a personagem fique imune a colisões que causam dano e consequente diminuição no número de Heart disponível durante determinado período de tempo.
- DeathBlock: bloco especial que causa dano por colisão e consequente diminuição no número de Heart disponível para a personagem.



Figura 4: Power-Ups e Power-Downs.

3.3 Níveis

Nick's Hardest Adventure conta com 3 níveis com diferentes temáticas, que apresentam grau de dificuldade e complexidade crescentes.

O nível 1 apresenta como temática o deserto. O nível 2, como temática a floresta e o nível 3, terras geladas.

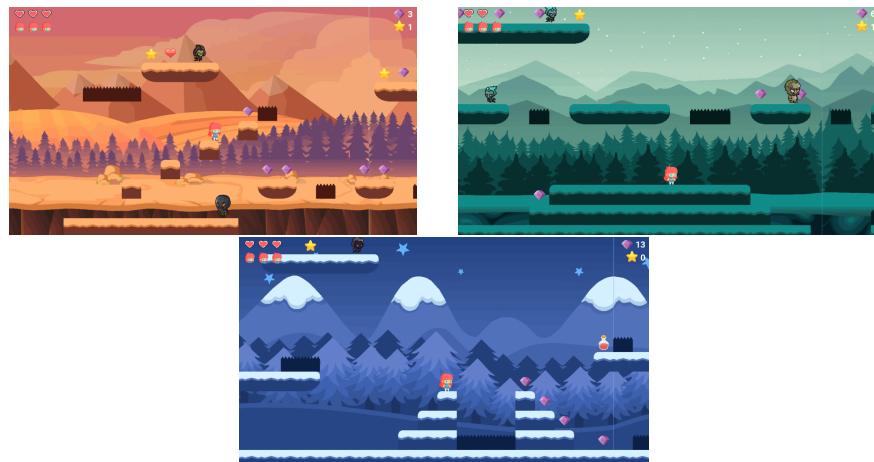


Figura 5: Níveis do jogo.

Cada um dos níveis apresenta inimigos de estéticas condizentes com cada um dos temas, sendo 2 tipos disponíveis, um com *normal size*, ou seja, inimigo de tamanho normal e com *large size*, inimigo com tamanho maior, porém ambos apresentam a mesma mecânica de derrota, já explicada acima.



Figura 6: Inimigos do jogo.

4 Elementos presentes no código no repositório do GitHub

O jogo foi implementado em 3 etapas. Em um primeiro momento foi construído o jogo em função das mecânicas básicas do herói presente no jogo, criando assim o personagem, os blocos no qual o boneco deve se movimentar (tiles), o background e as próprias interações entre estes, fazendo com que o boneco possuísse mecânicas de movimento, ajustando assim a física do jogo.

Posteriormente, foram implementados os sistema de vida, no qual o personagem pode morrer pela alteração da variável hearts, bem como um sistema de score, no qual o personagem pode coletar e assim fazer o jogo desafiador. Ademais, foram implementadas outras funcionalidades, como a passagem de nível e também a própria finalização de jogo, alterando a classe do jogo e a classe do próprio Level. Diante disso, pode-se construir os níveis de maneira mais fácil usando um arquivo json que basicamente configura os aspectos dos níveis em questão e após a variação do índice na lista de níveis, foi possível assim posicionar os blocos, itens e os próprios inimigos a partir de um grid de eixo x e y com variáveis especificadas em lista de par ordenado dentro do arquivo json. Logo, nessa segunda etapa o jogo já estava bem jogável e apresentava mecânicas sólidas de um jogo de plataforma.

Por último, na entrega final, foram trabalhados os detalhes do jogo e o seu próprio propósito, ou seja, a construção dos níveis, ajustando a dificuldade e tornando-o interessante de se jogar, vale salientar aqui que isso necessitou de uma implementação bloco por bloco, devido a impossibilidade de aleatoriedade. Ademais, foram trabalhados detalhes envolvendo a implementação da animação de personagem, construção de menu e mecânicas como o pause no jogo. Além de claro, finalização da estrutura corrigindo bugs e outros problemas presentes principalmente na animação e movimentação de bonecos com outros itens do cenário

A arquitetura presente consistiu basicamente em um arquivo denotado por "our game.py" que possui todo o código do jogo e dois arquivos "main.py" e "button.py" que fazem a construção do menu do jogo, bem como os botões presentes neste. Além disso, fez-se uso de diversas pastas para o uso dos sprites dentro do jogo, todas dentro da pasta denominada "assets", enquanto os níveis em arquivo json ficaram presente na pasta denotada por "levels". Em "assets" os nomes são intuitivos, e mostram os sprites usados nas animações do herói e dos inimigos, bem como ilustração de blocos, backgrounds e itens, e por fim também pode-se ver os sons, fontes e outras propriedades de design implementadas separadas por pasta.

As classes do jogo se resumem a:

- Entity : Entidade que basicamente representa tudo no jogo, aqui é aplicado gravidade e coordenadas x e y para posicionamento do objeto no grid imaginário

construído para cada nível.

- Block : Os próprios tiles do jogo, em que basicamente possuem diversos tipos, representando o centro, topo, direita ou esquerda de uma plataforma.
- DeathBlock: Um block que possui uma função associada de dano a ser aplicada caso o personagem encoste nele.
- Character: O próprio herói, que possui todas as suas imagens de animações em listas, além de diversas funções e variáveis associadas a sua implementação. Aqui pode-se dar destaque as função de movimento do personagem em relação aos blocos (sem colidi-los) e a interação com outras entidades do jogo, processamento de diamantes para score por exemplo, além de processamento de inimigos, gerando dano caso encoste neles. Por fim, pode-se usar funções de update e respawn para manter o jogo atualizado constantemente com as variáveis linkadas ao personagem principal.
- Coin: Coletável que contabiliza algum valor próprio. Deu-se esse nome pois inicialmente queríamos usar a mecânica de coins, mas depois optou-se por usar duas contagens de valor diferentes.
- Diamond: Herda de Coin a contabilização e tem como objetivo setar o score do player, com função de seu processamento específica para isso.
- Star: Herda de Coin a contabilização e tem como objetivo fazer o player coletar todos presentes em cada nível, dificultando assim o jogo, com função de seu processamento específica para isso.
- OneUp: Basicamente parte do sistema de vidas do personagem, que começa com 3 e vai perdendo aos poucos no jogo, representando o coletável que aumenta em um a vida.
- Hearth: Basicamente parte do sistema de corações do personagem, que quando se perde todos se perde uma vida, representando o coletável que aumenta em um o número de corações.
- Potion: Coletável que proporciona um tempo de invencibilidade ao jogador por uma função associada a própria variável de invencibilidade do herói, que já tinha sido implementada para ajudar na contabilidade de danos dos inimigos.
- Flag: Necessita de uma classe separada (podia ser só um tipo de bloco) pois marca o final do nível e portanto possui uma função de avanço de nível associada a este que inclusive se comunica bem com a class Level e Game.

- Enemy: Muito semelhante à classe Character, com várias funcionalidades implementadas de movimento e animação, aqui resta salientar que possui funções apenas associadas a própria morte e interação com o personagem, já que não tem coletável. Ademais, a única diferença notável é que a movimentação precisou ser um pouco diferente por sua automatização e também para fazer com que o inimigo não caísse de alguma plataforma, ajustando os limites de seu retângulo com os limites dos blocos.
- Bear: Herda inimigo e possui animação própria, além de mecanismo que seleciona o design para cada fase.
- Monster: Herda inimigo e possui mesmas funcionalidades do Bear. Representa um inimigo diferente, até porque a ideia inicial era fazer este especificamente atirar coisas no herói, mas infelizmente isso não foi implementado.
- Level: Herda inimigo e possui animação própria, além de mecanismo Possui a principal funcionalidade de ler o arquivo json que seta o grid imaginário e a partir do arquivo json posicionar cada objeto, fazendo realmente a parte de programação orientada a objetos. Aqui todos as listas de objetos são setadas por vazias e são preenchidas com a leitura do arquivo, colocando a imagem do tamanho apropriado ao grid, bem como background e coisas específicas. Ademais, tem função de reset para cada fase passada poder ser avançada por outra.
- Game: É o que roda os eventos do jogo em si, apresentando diversos estados de jogo denotados por JOGANDO, PAUSADO, COMPLETO, INCOMPLETO, START e assim sucessivamente. Esses status orientam o que deve ocorrer na interface do jogo, com o auxílio de funções postas dentro dessa classe relacionadas a display de mensagens. O processamento de eventos por sua vez origina a necessidade da função draw que faz os eventos virarem ilustração, tudo isso considerando o status atual do jogo e teclas (eventos) pressionadas ou clicadas. Por fim, mostra-se necessário função de início, reset e update adequadas a cada estágio do jogo e um loop que permite que o jogo rode tranquilamente.

Este é o jogo, e a sua implementação fez de nosso estudo de Programação Orientada a Objetos algo muito bacana e posto bem em prática.