Apresentação das Mecânicas de um Jogo Desenvolvido com Arcade

Vítor Augusto Ueno Otto
IFC Campus BlumenauBlumenau, Santa Catarina
vitoruenootto@gmail.com

Alisson Reinaldo Flores
IFC Campus Blumenau
Blumenau, Santa Catarina
alisonreinaldof21@gmail.com

Manuela Helena Weidmann
IFC Campus BlumenauBlumenau, Santa Catarina
manuelahw12@gmail.com@gmail.com

Kauan Claudio Elias
IFC Campus BlumenauBlumenau, Santa Catarinakceliass@gmail.com

Ricardo de la Rocha Ladeira
IFC Campus Blumenau
Blumenau, Santa Catarina
ricardo.ladeira@ifc.edu.br

Aldelir Fernando Luiz
IFC Campus BlumenauBlumenau, Santa Catarinaaldelir.luiz@ifc.edu.br

Adriano Pessini
IFC Campus Blumenau
Blumenau, Santa Catarina
adriano.pessini@ifc.edu.br

ABSTRACT

The purpose of this paper is to describe the mechanics and logic of Bombanimal's creation, a game inspired by the classic Bomberman and created using the Python programming language combined with the 2D game development library Arcade. The game has unique features such as the skills of the characters and its visual theme. As such, the paper discusses how the game was implemented and what Arcade classes, methods, and features were used, in order to encourage other programmers to use Arcade on their own games.

KEYWORDS

Game Development, Bomberman Mechanics, Python, Arcade

1 INTRODUÇÃO

Jogos eletrônicos são uma forma relativamente moderna de entretenimento, surgida no final da década de 50 com o uso de um osciloscópio, instrumento longe de ser considerado um videogame [1]. Atualmente, são compostos não apenas por diversos gêneros e estilos, mas também na forma como de fato foram planejados e implementados, e quais tecnologias e metodologias foram empregadas.

Tendo isso em mente, Bombanimal foi criado: um jogo para dois jogadores simultâneos em um mesmo dispositivo, seguindo um estilo bem conceituado da franquia Bomberman. Para sua implementação optou-se pelo uso de ferramentas simples, gratuitas e flexíveis: linguagem python (com a biblioteca Arcade) e os serviços para criação de desenhos pixilart e para hospedagem de códigos Github.

Dessa maneira, o presente artigo busca demonstrar as peculiaridades do Bombanimal no âmbito de sua mecânica, apresentando a lógica que foi empregada para a resolução das principais desafios de desenvolvimento. Para isso pretende-se demonstrar o nível de lógica de programação necessária para a criação de um jogo simples e quais métodos e funções da biblioteca Arcade mais foram utilizadas nesse processo.

2 MATERIAIS E MÉTODOS

antes da fase de implementação, descreveu-se as características que o Bombanimal possuiria no GDD (*Game Design Documet*), a fim de evitar mudanças bruscas de objetivo e favorecer a criação de cronogramas de desenvolvimento. Optou-se por utilizar a linguagem Python [2] por ser de fácil uso, legível e gratuita, aliada à biblioteca para desenvolvimento de jogos 2D Arcade [3], que foi criada em 2017 e que possibilita a produção de jogos de forma simples. As artes, no estilo pixel art, foram desenhadas no site Pixilart [4] e o código fonte, bem como os diagramas e GDD's foram hospedados no GitHub[1] [5], site conceituado para compartilhamento de códigos e arquivos.

3 DESENVOLVIMENTO

As partidas de Bombanimal são disputadas por dois jogadores que podem escolher um dentre os quatro personagens disponíveis (Figura 1): lebre, panda, arara-azul ou pinguim. Cada um deles apresenta uma habilidade única que influencia na *gameplay*. A lebre começa com 15% mais velocidade de movimento, o pinguim pode posicionar uma bomba a mais simultaneamente, o panda pode empurrar bombas sem o item "mão de primata"e a arara-azul pode pular um bloco a cada 5 segundos.

Após escolher o personagem, a partida é iniciada (Figura 2), com cada jogador sendo posicionado em um dos extremos do mapa, oposto um ao outro. A partir daí, o objetivo do jogo é eliminar o adversário e ser o último sobrevivente. Para Cumprir essa missão, cada jogador pode posicionar bombas que explodem em forma de cruz, destruíndo os blocos destrutíveis ou jogadores que estiverem em seu raio de explosão. Os blocos destrutíveis, por sua vez, tem chance de gerar itens que melhoram os atributos do personagens: bomba extra, fogo, vento, mão de primata e capacete de coco.

3.1 Mapa e Blocos

O mapa completo do jogo ocupa todos os 800x600 *pixels* da tela e possui um visão superior. Para o programa, ela é considerada uma

[1] https://github.com/vitorueno/BombAnimal

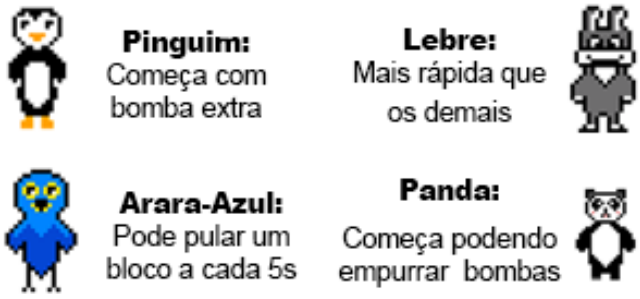

Figura 1: personagens do Bombanimal e suas habilidades únicas

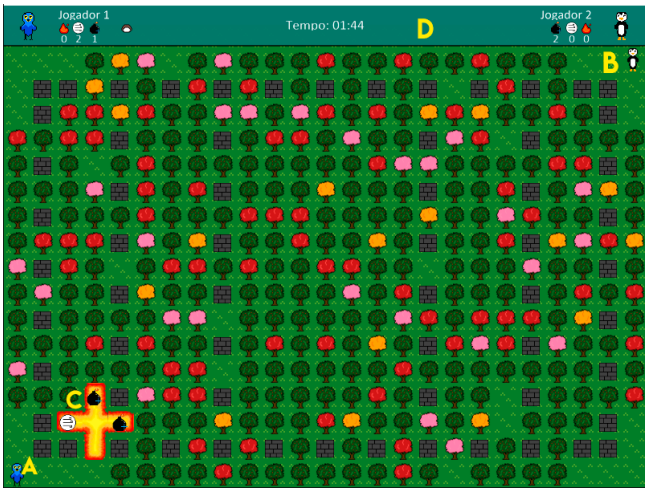

Figura 2: Partida do Bombanimal

matriz, ou uma lista de listas, onde seus elementos descrevem qual tipo de bloco pode ser posicionado em cada ponto do mapa. Todos eles são considerados sprites para a biblioteca de criação de jogos, herdando características que facilitam seu gerenciamento da classe `Sprite` do Arcade. Dessa forma, todos possuem uma foto, uma coordenada (x e y), limites de colisão e métodos predefinidos que podem ser reescritos de acordo com as necessidades: `update` (atualizar ações do sprite), `on_draw` (como ele será desenhado na tela), `on_key_press` e `on_key_release` (teclas pressionadas e soltas), dentre outras.

A grama faz parte do cenário do jogo, portanto o primeiro a ser desenhado na tela. As paredes são classificadas como blocos indestrutíveis e sua posição é fixa. Por fim, as árvores compõem os elementos destrutíveis do jogo, o tipo de árvore é gerado aleatóriamente e isso influência a chance de gerar itens quando são destruídas. De acordo com o grau de raridade da árvore sorteada na geração do mapa, ela terá maior ou menor chance de dar origem a itens (Figura 3).

Após terem sido originados, uma `SpriteList` (lista de gerenciamento de sprites do Arcade) é criada para cada elemento do mapa no método `setup` do jogo (método de associação de valores inicias às variaveis já inicializadas), sendo posterioremente utilizados nos métodos de `update` e `on_draw` que tratam colisões e os desenham na tela, respectivamente.

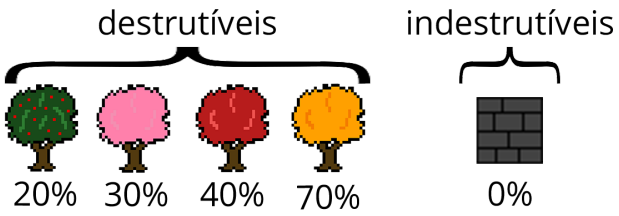

Figura 3: Blocos do Bombanimal com sua probabilidade de gerar itens

3.2 Movimentação e Colisões

Os personagens, que também são sprites, se movimentam quando as respectivas teclas são pressionadas e essa verificação ocorre no `on_key_press` e `on_key_release`. Nesses métodos, cada tecla de movimentação é associada a uma variável `booleana` que representa uma direção (cima, baixo, direita e esquerda), recebendo o valor verdadeiro (`True`) apenas quando pressionadas. Neste momento, a posição do personagem é alterada positivamente ou negativamente nos eixos x e y, de acordo com a tecla pressionada, na taxa de sua `velocidade`, um de seus atributos. Portanto, pressionar para cima ou para baixo alterará a posição no eixo y, enquanto para esquerda ou direta altera a posição no eixo x. Enquanto a tecla permanecer pressionada, o valor da velocidade dele continuará sendo adicionada ou reduzida da sua posição x (atributo `center_x`) e y (`center_y`) .

Se no percurso do jogador, um bloco é encontrado, seja ele destrutível ou indestrutível, o jogador tem seu deslocamento interrompido. Para o sistema, o que realmente acontece é uma verificação nos limites de ambos os `sprites`: se o limite direito, esquerdo, inferior ou superior do jogador estiverem invadindo as coordenadas do bloco, a posição central do personagem é corrigida para o mínimo possível antes dessa invasão acontecer. Essa verificação deve ser constante para dar a impressão de que nenhuma parede, árvore ou limite do mapa foi invadido.

3.3 Bomba e Explosões

Assim que a tecla responsável pelo posicionamento da bomba é pressionado no método `on_key_press`, um Sprite de bomba é criada e associada à `Spritelist` de bombas do jogo. Nesse momento, um *timer* que é incrementado no método `update`, sendo checado ciclicamente até atingir a marca de aproximadamente três segundos, momento em que a bomba é eliminada do jogo através do método `kill`, dando lugar a uma explosão (Figura 4).

As bombas, assim como os blocos, também são consideradas obstáculos para o jogador, mas sua verificação de colisão é realizada de forma distinta, pois leva em conta que a colisão só pode iniciar depois que o jogador que a posicionou se distanciou o suficiente dela. Caso o jogador possua o item mão de primata, que o permite empurrar bombas, este pode modificar a posição das mesmas.

As explosões, ao contrário dos outros sprites, são compostos por vários componentes, que quando combinados formam a cruz característica. Para poupar processamentos desnecessários, todas

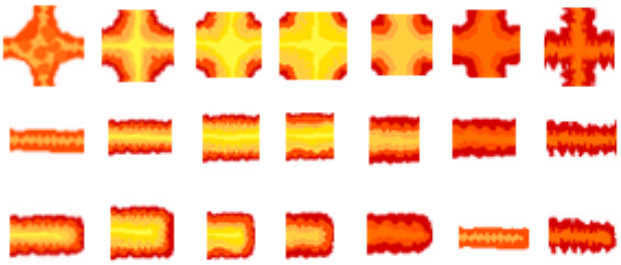

Figura 4: Imagens que compõem uma explosão no Bombanimal

as imagens para a animação da explosão são carregadas no método `setup`, e então passadas como parametro da criação das bombas. Dessa forma, quando a explosão se inicia, as imagens dela são alteradas rapidamente num determinado intervalo para dar impressão de uma animação, e quando a último conjunto de imagens acaba, a explosão some (é eliminada com o método `kill`).

Nesse meio tempo, caso um jogador ou bloco destrutível entrar no limite da explosão, ele também receberá o método `kill`. Se for um bloco ele poderá iniciar o processo de geração de itens, e se for um personagem sem o item capacete de coco, que garante imunidade a uma bomba, a partida se encerrá, declarando que o outro jogador foi o vitorioso.

3.4 Itens

Após explodido, um dos cinco itens podem ser gerados (Figura 5): mão de primata que permite empurrar cada bomba uma vez, o capacete de coco que protege o personagem da próxima explosão lhe dando imunidade por um segundo e meio, bomba extra aumenta o limite de bombas que podem ser posicionadas simultaneamente, fogo que aumenta o raio da explosão em um (a partir de seu centro) e o vento que aumenta a velocidade de movimento do personagem em 15%. Além das probabilidades que cada árvore tem de gerá-los, existem também chances distintas para cada uma das melhorias, sendo o vento o mais comum e o capacete de coco o mais raro.

Figura 5: Itens do Bombanimal com sua probabilidade de ser gerado em relação aos demais

Após coletado, as melhorias afetam os atributos do personagem ou acionam métodos específicos para ele: o vento incrementa o atributo `velocidade`, o fogo a `força` e o bomba extra a `numero_bombas`; o capacete de coco, por sua vez, altera a funcionalidade do método `verificar_vida` que é acionado quando uma colisão entre jogador e explosão ocorre, tornando o estado do atributo `imortal` verdadeiro, fazendo com que o método `atualizar_tempo_imune` seja ativado, até a imunidade acabar, quando o jogador se torna vulnerável novamente. Por fim, o mão de primata modifica as verificações de colisão da bomba, localizadas na classe do jogo.

3.5 Telas e Botões

Cada tela de menu do Bombanimal é uma classe e possui algumas características em comum (Figura 6): as imagens de fundo e os títulos são na verdade `sprites`, possuindo uma imagem (de fundo ou de texto) e coordenadas de onde ficarão na tela. Apesar disso, nenhuma lógica ocorre na atualização desses objetos, sendo apenas desenhados na tela com o método `draw`.

Os botões, por outro lado, são objetos da classe `Botão`, que apresenta a lógica de clique nos métodos `on_mouse_press` e também no `on_mouse_release`. Para criar um botão é necessário apenas de uma imagem que o representará, o tamanho da largura e altura da área de clique e um número ou texto que será retornado quando ele for clicado. Através da checagem da posição `x` e `y` do clique do usuário, é possível determinar se ele acertou um botão, e caso isso ocorra, o valor associado a ele é retornado, possibilitando a execução da ação desejada, tal como alterar o estado da partida (ir do menu para a seleção de personagens ou tela de opções, por exemplo).

Figura 6: Menu do Bombanimal construído com Arcade

4 AVALIAÇÃO

Durante a apresentação do projeto na Mostra de Ensino, Pesquisa, Extensão e Cidadania (MEPEC) do IFC campus Blumenau, foram coletadas seis avaliações do Bombanimal através de um formulário que apresenta perguntas de múltipla escolha e espaço para comentários, totalizando nove campos.

Das respostas coletadas, 83% avaliaram o jogo com uma nota igual ou superior a oito e destes, 50% deram nota máxima. Além disso, 66,7% disseram que voltariam a jogá-lo novamente.

Apesar da avaliação positiva, 66,7% dos participantes relataram algum tipo de *bug*, referentes a travamentos ou dificuldade de movimentação. Das críticas coletadas, a principal reclamação foi a chance de surgimento de itens excessivamente alta, que assim como a díficil movimentação foram corrigidas.

Por fim, os elementos do jogo mais aprovados foram as artes, as habilidades únicas dos personagens e a navegação nos menus, que foram todos marcados em 83,3% das respostas. Por sua vez, o menos marcado foi a chance de surgimento de itens, com apenas 50% das respostas.

Desta forma, por meio das avaliações coletadas, pôde-se realizar correções que foram aplicadas na versão final do jogo.

5 CONSIDERAÇÕES FINAIS

Por meio do presente trabalho, notou-se que é simples criar um jogo no estilo Bomberman com Python e Arcade, tendo em vista que as plataformas utilizadas para se criar o Bombanimal são gratuitas e os únicos requisitos são conhecer lógica de programação, matemática e Python. A partir dos conhecimentos adquiridos na criação de um jogo,aprimora-se habilidades de programação.

Percebeu-se que as mecânicas de jogo envolvem conceitos matemáticos, tais como coordenadas no plano cartesiano e matrizes, que se traduzem, no código, em uso de laços de repetição para percorrer listas. Não obstante, conceitos físicos, linguísticos e habilidades artísticas também foram postos em prática, além da engenharia de software.

Com relação aos trabalhos futuros para o jogo, pretende-se colher novas avaliações e realizar correções de eventuais *bugs*. Além disso, abre-se a possibilidade de criação de um instalador, adição de um novo sistema de geração de mapas e o estudo para adicionar mais do que apenas dois jogadores nas partidas. Outras possibilidades de modificação existirão, tendo em vista que o jogo sempre poderá ser melhorado.

Portanto, a criação de jogos se mostrou uma forma divertida de colocar em prática o desenvolvimento nos moldes da engenharia de software. Os procedimentos escolhidos permitiram agilidade para o projeto e para isso foi fundamental sua fase de especificação. Finalmente, o uso da biblioteca Arcade foi decisivo para atingir os objetivos do presente trabalho, em virtude de sua simplicidade e de sua documentação satisfatória.

REFERÊNCIAS

[1] Barboza, E. F. U. e Silva, A. C. de A. A evolução tecnológica dos jogos eletrônicos: do videogame para o newsgame. *5º Simpósio Internacional de Ciberjornalismo*, pages 1–16, 2014. URL http://www.ciberjor.ufms.br/ciberjor5/files/2014/07/eduardo.pdf.
[2] About python. URL https://www.python.org/about/.
[3] The python arcade library. URL http://arcade.academy/.
[4] Create and share art. URL https://www.pixilart.com/.
[5] Built for developers. URL https://github.com/.