**Projeto (estrutura) do programa**

O projeto foi desenvolvido por completo em Java com o auxílio de um framework específico para o desenvolvimento de jogos: libGDX. Desenvolvido pelo grupo de desenvolvimento Bad Logic Games, sendo de baixo nível, gratuito, de código-fonte aberto e licenciado com Apache2, ele permite que qualquer desenvolvedor o utilize gratuita e livremente.

Ele possui suas próprias classes e interfaces prontas para serem herdadas ou implementadas e, assim, dar início ao projeto. Ele usa preferencialmente a JDK (Java Development Kit) 8 e a ferramenta de montagem de projetos Gradle, responsável por melhorar a performance da aplicação e do desenvolvimento.

Para se ter uma iniciação de projeto libGDX, é necessário executar a ferramenta de criação própria e especificar todas as informações de criação de projeto, conforme a figura abaixo:



Para se ter um projeto mínimo e funcional, é necessário ter uma classe que herda uma classe-mãe chamada ApplicationAdapter, o exemplo abaixo demonstra como abrir uma janela disponibilizando uma imagem simples:



*SpriteBatch* é o tipo de dado da libGDX capaz de inicializar uma instância de tela. O método sobrescrito *create* referencia os principais componentes da aplicação e o método sobrescrito *render* especifica as cores e as configurações de renderização de imagem com as funções da classe Gdx. Para disponibilizar a imagem visualmente, inicia-se *batch*, usa-se seu método *draw* com os respectivos parâmetros, e após isso, deve-se finalizá-lo.

Com base neste funcionamento, nosso grupo entrou em um consenso de desenvolver um jogo de plataforma baseado no clássico *Super Mario Bros* com uma temática sustentável a respeito das abelhas.

A organização dos arquivos de classe consiste em divisão em pacotes (pastas), cada um preservando seu significado:

Items: pacote que segura uma classe abstrata Item, uma classe de definição ItemDef e uma classe física chamada Mushroom (cogumelo), em referência ao crescimento do personagem principal ao entrar em contato com um objeto desta classe.

Scenes: pacote responsável por controlar cenas específicas do jogo, segurando apenas um arquivo de classe chamado Hud, que controla o funcionamento das informações que aparecem no jogo principal, como a quantidade de pontos, o nome do mundo e o tempo para completar a fase.

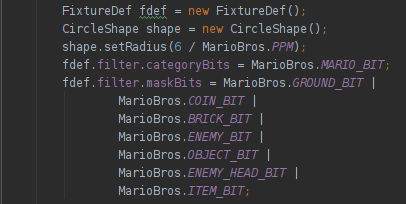
Screens: pacote responsável por controlar a tela principal do jogo, o conteúdo completo que irá aparecer na janela de jogo. Nele se encontram as classes Credits (créditos), GameOverScreen (fim de jogo), MainMenu (menu principal) e PlayScreen (a classe do jogo em si).

Sprites: pacote responsável por armazenar as principais entidades do jogo, como o personagem principal (representado na classe Mario, que resulta na abelha que o usuário controla), a classe abstrata para a montagem de inimigos (Enemy), os inimigos principais (Goomba e Turtle), que resultam nas entidades empresário e urso e objetos interativos, como os blocos que invocam pontos e os que invocam o item Mushroom.

Tools: possui as classes de controle do projeto, como B2WorldCreator, responsável pela instanciação de todos os componentes do jogo utilizando-se da biblioteca Box2D (responsável pela administração de componentes nas duas primeiras dimensões) e WorldContactListener, classe responsável por controlar todas as interações entre as entidades e objetos interativos do pacote Sprites.

Além dos pacotes, temos a classe principal MarioBros e a classe executável com o método main DesktopLauncher, que reside dentro do pacote source, enquanto que todos os componentes mencionados anteriormente residem dentro do pacote core.

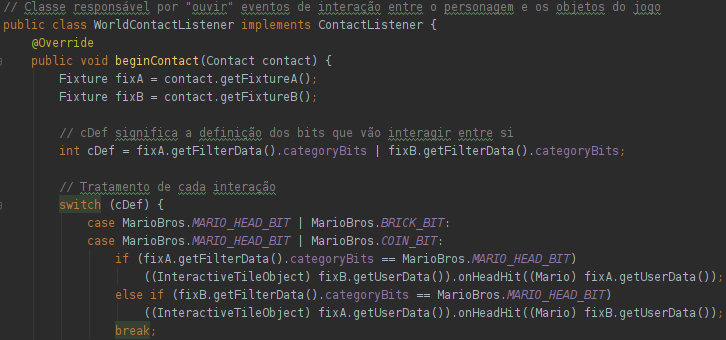
Para a interação entre entidade e objeto, foi utilizado um recurso de linguagem chamado operação bit a bit, que consiste em comparar logicamente valores de tipos de menor memória (bit, byte e short) para retornar um valor específico. Para cada entidade foi passado seu próprio identificador bit e todos os outros com os quais ele pode interagir. O exemplo abaixo demonstra isso na classe Mario:



FixtureDef é o tipo de classe para se criar uma fixação, em outras palavras, para determinar os limites de interação e com quem aquele objeto pode interagir. CircleShape é o tipo de classe para manipular o formato de uma fixação. O método setRadius determina o raio da circunferência. E, finalmente, o fragmento de código em questão, filter.categoryBits é o atributo ao qual o

objeto da classe Mario se refere para interagir com outros objetos e entidades, e filter.maskBits especifica exatamente com quais objetos ele pode interagir.

Já, na classe WorldContactListener, a interação entre a classe Mario e a classe Brick/Coin seria tratada da forma abaixo:



Em primeira mão, deve-se armazenar, no método principal beginContact, em duas variáveis as informações de ambos os objetos (fixA e fixB), com os getters de fixtureA e fixtureB. A variável cDef é o resultado do operador bitwise “ou”, representado por uma barra vertical, para indicar quais objetos estão interagindo entre si. Abaixo é realizado o tratamento de cada interação com um bloco de código de switch-case. No exemplo dado, o framework não possui uma ação específica de controle de variáveis, então em uma interação entre Mario e Brick, fixA poderia estar apontando para Mario, assim como fixB poderia estar apontando para Brick, e vice-versa, portanto é necessário tratar as duas ocasiões com uma sequência condicional if-else.

A realização da fase principal do jogo foi feita utilizando um software específico de criação de mapas chamado Tiled, capaz de organizar todos os elementos visuais e objetos de forma prática e simples, gerando, ao exportar, um arquivo de mapa com formato tmx (Translation Memory Exchange), um formato padrão XML aberto para realização de troca de tradução de memória. O libGDX possui uma classe especial para utilizar mapas com esse formato, que é a TmxMapLoader.

Todas as artes de jogo foram realizadas pelo nosso grupo e utilizadas no projeto com um arquivo .png e um arquivo .atlas traduzido a partir deste primeiro. Não só as artes, como também as animações, foram construídas do mesmo método em forma de frames. O framework possui tipos específicos de classe capazes de lidar com essas formas, que são TextureRegion e Animation<TextureRegion> (Animation de tipo TextureRegion).

Há ainda mais funcionalidades que o framework libGDX pode oferecer, como o tratamento de objetos em 3D (terceira dimensão), animações complexas, exportação de projeto para plataformas celulares, entre outros, que não julgamos necessário implementar no projeto.

Abaixo estão alguns exemplos do projeto ao seu final:

