libGDX

https://github.com/libgdx/libgdx/wiki/

Rafael Vieira Coelho

rafael.coelho@farroupilha.ifrs.edu.br

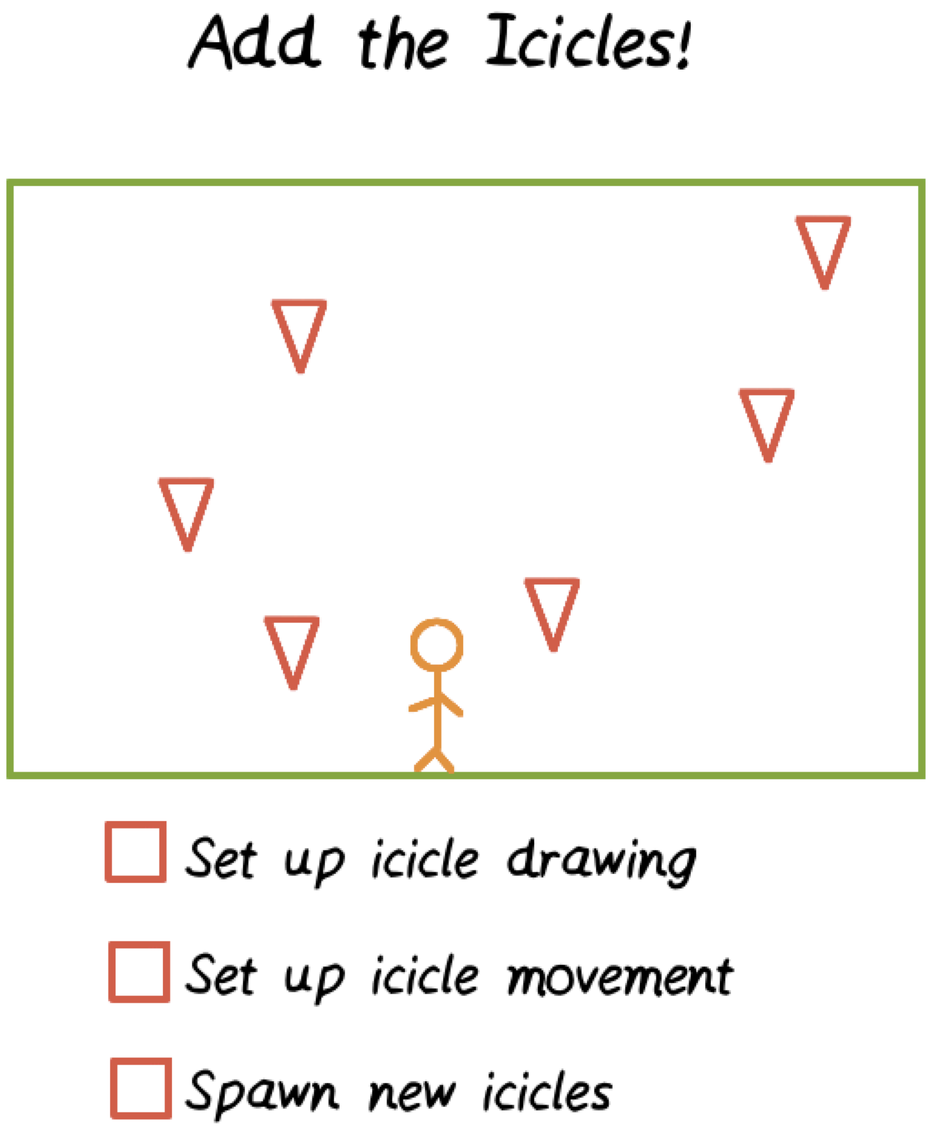
Parte 6 - Jogo Completo

- Versão 1 - Construção do Projeto
- Versão 2 - Desenha icicle
- Versão 3 - Desenha Jogador
- Versão 4 - Controle de Teclado (setas)
- **Versão 5 - Adiciona Icicles**
- Versão 6 - Remove Icicles que somem da tela
- Versão 7 - Detecta Colisão
- Versão 8 - Adiciona o HUD
- Versão 9 - Adiciona níveis de dificuldade
- Versão 10 - Adiciona seleção de nível de dificuldade

https://github.com/libgdx/libgdx/wiki/

Como Adicionar Vários Blocos de Gelo?

1. Precisamos desenhar icicles
2. Definir o movimento dos icicles
3. E criar novos icicles da parte superior da tela



Classe Constants.java

- Adicionamos uma constante para definir a aceleração dos icicles (objeto Vector2)
- Adicionamos uma constante para definir a quantidade de icicles devem ser criados por segundo.

Constants.java | IciclesScreen.java | Icicle.java | Icicles.java

```
6
7   public class Constants {
8       public static final float WORLD_SIZE = 10.0f;
9       public static final Color BACKGROUND_COLOR = Color.BLUE;
10
11      public static final float PLAYER_HEAD_RADIUS = 0.5f;
12      public static final float PLAYER_HEAD_HEIGHT = 4.0f * PLAYER_HEAD_RADIUS;
13      public static final float PLAYER_LIMB_WIDTH = 0.1f;
14      public static final int PLAYER_HEAD_SEGMENTS = 20;
15      public static final Color PLAYER_COLOR = Color.BLACK;
16      public static final float PLAYER_MOVEMENT_SPEED = 10.0f;
17
18      public static final float ACCELEROMETER_SENSITIVITY = 0.5f;
19      public static final float GRAVITATIONAL_ACCELERATION = 9.8f;
20
21      public static final float ICICLES_HEIGHT = 1.0f;
22      public static final float ICICLES_WIDTH = 0.5f;
23      public static final Color ICICLE_COLOR = Color.WHITE;
24
25      // TODO: Add constant for icicle acceleration
26      public static final Vector2 ICICLES_ACCELERATION = new Vector2( x: 0,  y: -5.0f);
27
28      // TODO: Add constant for icicle spawns per second
29      public static final float ICICLE_SPAWNS_PER_SECOND = 10.0f;
30  }
```

Classe Icicle.java

- Definimos um objeto Vector para representar a velocidade do icicle
- E inicializá-lo no construtor
- No método update(), precisamos atualizar a velocidade com base na aceleração definida previamente em Constants.java
- E modificar a posição com base nesta velocidade.

Icicle.java | Icicles.java | IciclesScreen.java

```
6   public class Icicle {
7       public static final String TAG = Icicle.class.getName();
8       Vector2 position;
9
10      // TODO: Add Vector2 for velocity
11      Vector2 velocity;
12
13      public Icicle(Vector2 position) {
14          this.position = position;
15          // TODO: Initialize velocity
16          this.velocity = new Vector2();
17      }
18
19      public void update(float delta) {
20          // TODO: Update velocity using icicle accelration constant
21          velocity.mulAdd(Constants.ICICLES_ACCELERATION, delta);
22
23          // TODO: Update position using velocity
24          position.mulAdd(velocity, delta);
25      }
26
27  @   public void render(ShapeRenderer renderer) {
28          renderer.triangle(
29                  position.x, position.y,
30                  x2: position.x - Constants.ICICLES_WIDTH / 2,
31                  y2: position.y + Constants.ICICLES_HEIGHT,
32                  x3: position.x + Constants.ICICLES_WIDTH / 2,
33                  y3: position.y + Constants.ICICLES_HEIGHT
34          );
```

Classe Icicles.java

- Criamos um ArrayList de objetos da classe Icicle.java: **icicleList**
- Inicializamos este ArrayList no método **init()**
- E no método **update()** criamos um fator randômico que atualiza com base na constante definida em Constants.java
- Caso a condição seja verdadeira, criamos um objeto **Vector2** em uma posição aleatória dentro do mundo do jogo
- E atualizamos todos icicles do ArrayList **icicleList**

Icicles.java | IciclesScreen.java

```
9   public class Icicles {
10      // TODO: Add an array of icicles and a viewport
11      Array<Icicle> icicleList;
12      Viewport viewport;
13
14      public void init() {
15          // TODO: Initialize the array of icicles
16          icicleList = new Array<Icicle>( ordered: false,  capacity: 100);
17      }
18
19      public void update(float delta) {
20          // TODO: Replace hard-coded spawn rate with a constant
21          if (MathUtils.random() < delta * Constants.ICICLE_SPAWNS_PER_SECOND) {
22              // TODO: Add a new icicle at the top of the viewport
23              //  at a random x position
24              Vector2 newIciclePosition = new Vector2(
25                      x: MathUtils.random() * viewport.getWorldWidth(),
26                      viewport.getWorldHeight()
27              );
28              Icicle newIcicle = new Icicle(newIciclePosition);
29              icicleList.add(newIcicle);
30          }
31          // TODO: Update each icicle
32          for (Icicle icicle : icicleList) {
33              icicle.update(delta);
34          }
35      }
```

Classe Icicles.java

- No método que desenha o Icicle (**render()**), definimos a cor com base na constante **ICICLE_COLOR**
- E redesenhamos todos os icicles do. ArrayList **icicleList**.

Icicles.java | IciclesScreen.java

```
public class Icicles {

    public void render(ShapeRenderer renderer) {
        // TODO: Set ShapeRenderer Color
        renderer.setColor(Constants.ICICLE_COLOR);

        // TODO: Render each icicle
        for (Icicle icicle : icicleList) {
            icicle.render(renderer);
        }
    }
```

Classe IciclesScreen.java

- Criamos um atributo com um objeto da classe **Icicles**
- E instanciamos ele no método **show()**
- Precisamos reiniciar os icicles caso seja redimensionada a tela (método **resize()**)

IciclesScreen.java

```
10  public class IciclesScreen implements Screen {
11      // TODO: Add an instance of Icicles
12      Icicles icicles;
13
14      @Override
15      public void show() {
16          iciclesViewport = new ExtendViewport(
17                  Constants.WORLD_SIZE,
18                  Constants.WORLD_SIZE
19          );
20          renderer = new ShapeRenderer();
21          renderer.setAutoShapeType(true);
22          player = new Player(iciclesViewport);
23          // TODO: Initialize icicles
24          icicles = new Icicles(iciclesViewport);
25      }
26
27      @Override
28      public void resize(int width, int height) {
29          iciclesViewport.update(width, height,  centerCamera: true);
30          player.init();
31          // TODO: Reset icicles
32          icicles.init();
33      }
```

Classe IciclesScreen.java

- Precisamos redesenhar os icicles no método **render()**

IciclesScreen.java

```
10  public class IciclesScreen implements Screen {
11      // TODO: Add an instance of Icicles
12      Icicles icicles;
13
14      @Override
15      public void show() {...}
26
27      @Override
28      public void resize(int width, int height) {...}
34
35      @Override
36      public void render(float delta) {
37          // TODO: Update Icicles
38          icicles.update(delta);
39          player.update(delta);
40
41          iciclesViewport.apply( centerCamera: true);
42          Gdx.gl.glClearColor(Constants.BACKGROUND_COLOR.r,
43                  Constants.BACKGROUND_COLOR.g,
44                  Constants.BACKGROUND_COLOR.b,  alpha: 1);
45          Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);
46
47          renderer.setProjectionMatrix(iciclesViewport.getCamera().combined);
48          renderer.begin(ShapeType.Filled);
49          // TODO: Render Icicles
50          icicles.render(renderer);
51          player.render(renderer);
52          renderer.end();
53      }
```