



libGDX

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

Rafael Vieira Coelho

rafael.coelho@farroupilha.ifrs.edu.br



Parte 6 - Jogo Completo

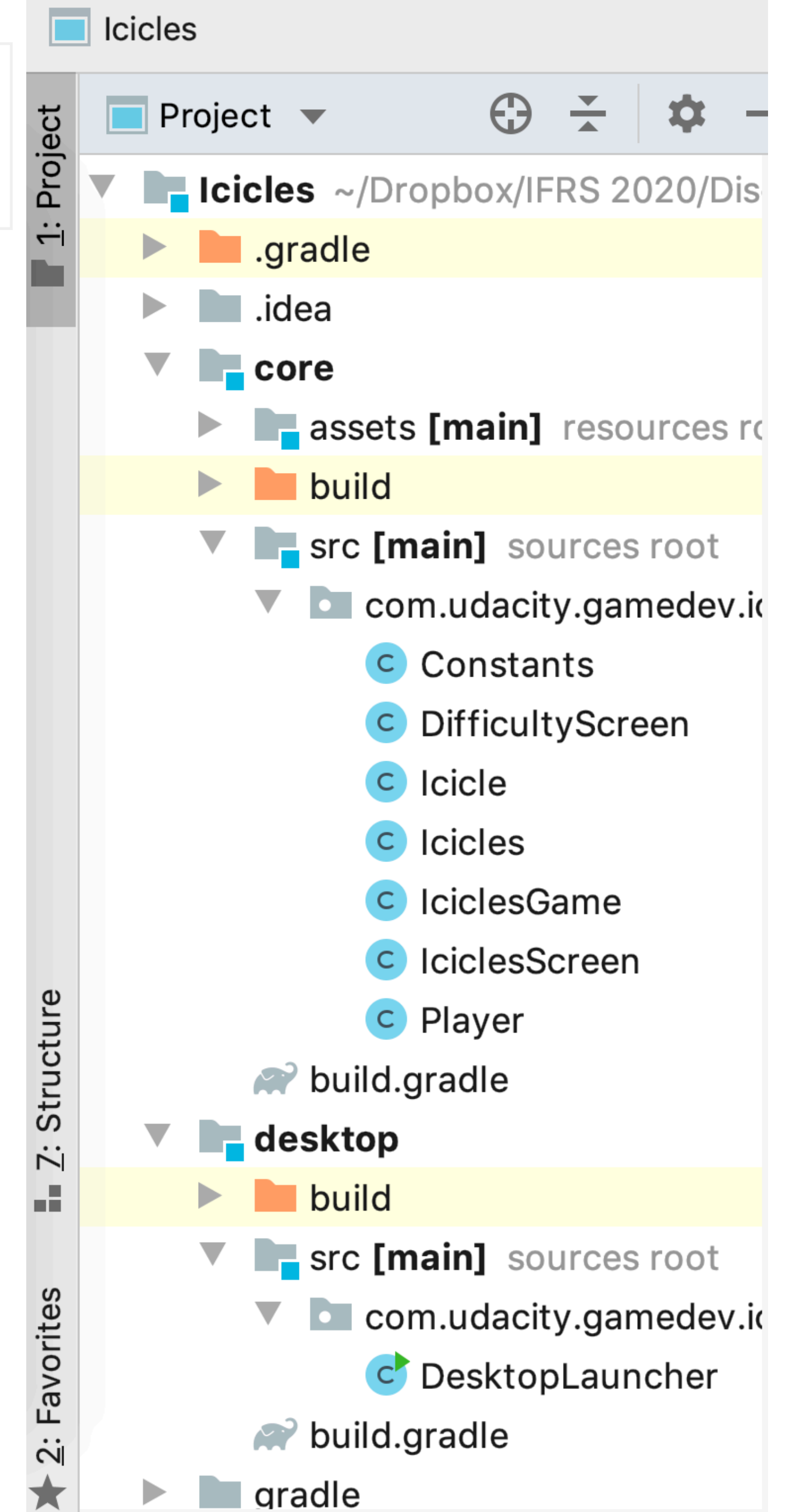
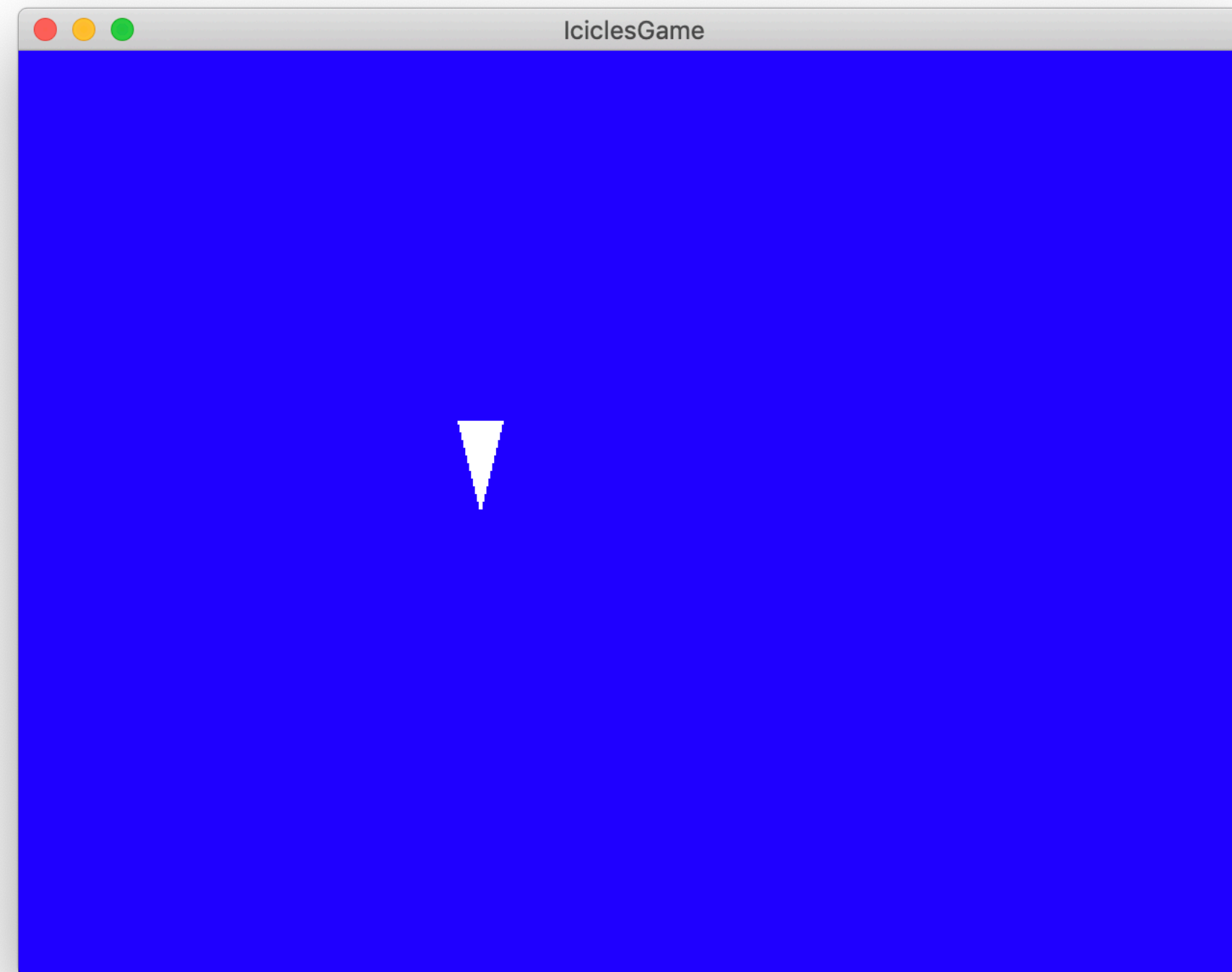
- Versão 1 - Construção do Projeto
- **Versão 2 - Desenha icicle**
- Versão 3 - Desenha Jogador
- Versão 4 - Controle de Teclado (setas)
- Versão 5 - Adiciona Icicles
- Versão 6 - Remove Icicles que somem da tela
- Versão 7 - Detecta Colisão
- Versão 8 - Adiciona o HUD
- Versão 9 - Adiciona níveis de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

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



Estado Atual do Projeto

- Desenharemos o icicle na tela



Classe Constants.java

- A classe Constants armazena valores fixos que representam propriedades do nosso jogo.

```
Constants.java ×
1 package com.udacity.gamedev.icicles;
2
3 import com.badlogic.gdx.graphics.Color;
4
5 public class Constants {
6
7     // TODO: Add a constant for the world size
8     public static final float WORLD_SIZE = 10.0f;
9
10    // TODO: Add a constant for the background color of the world
11    public static final Color BACKGROUND_COLOR = Color.BLUE;
12
13    // TODO: Add a constant for the height of the icicle
14    public static final float ICICLES_HEIGHT = 1.0f;
15
16    // TODO: Add a constant for the width of the icicle
17    public static final float ICICLES_WIDTH = 0.5f;
18
19    // TODO: Add a constant for the color of the icicles
20    public static final Color ICICLE_COLOR = Color.WHITE;
21 }
```

Classe ICicleGame.java

- Precisamos definir a tela inicial.

```
1    package com.udacity.gamedev.icicles;
2
3    import com.badlogic.gdx.Game;
4
5
6    public class IciclesGame extends Game {
7
8        //TODO: call setScreen() with a new IciclesScreen()
9        @Override
10       public void create() {
11           setScreen(new IciclesScreen());
12       }
13   }
```

Como Desenhar um Bloco de Gelo?

1. Precisamos definir o ShapeRenderer para que o objeto desejado possa ser renderizado (desenhado) na tela.
2. Depois definir uma forma de ver ele na tela (ExtendViewport), como uma câmera.
3. E desenhar o objeto.

The First Icicle



- ☐ Set up ShapeRenderer
- ☐ Set up ExtendViewport
- ☐ Draw an Icicle!

Classe ICicleScreen.java

1. Precisamos definir o **ShapeRenderer** para que o objeto desejado possa ser renderizado (desenhado) na tela.
2. Depois definir uma forma de ver ele na tela (**ExtendViewport**), como uma câmera.
3. E desenhar o objeto.

```
1  package com.udacity.gamedev.icicles;
2
3  import com.badlogic.gdx.Gdx;
4  import com.badlogic.gdx.Screen;
5  import com.badlogic.gdx.graphics.GL20;
6  import com.badlogic.gdx.graphics.glutils.ShapeRenderer;
7  import com.badlogic.gdx.graphics.glutils.ShapeRenderer.ShapeType;
8  import com.badlogic.gdx.math.Vector2;
9  import com.badlogic.gdx.utils.viewport.ExtendViewport;
10
11 public class IciclesScreen implements Screen {
12
13     public static final String TAG = IciclesScreen.class.getName();
14
15     // TODO: Add an ExtendViewport
16     ExtendViewport iciclesViewport;
17
18     // TODO: Add a ShapeRenderer
19     ShapeRenderer renderer;
20
21     // TODO: Add an Icycle
22     Icycle icicle;
```

Classe ICicleScreen.java




```
24  @Override
25  public void show() {
26      // TODO: Initialize the viewport using the world size constant
27      iciclesViewport = new ExtendViewport(Constants.WORLD_SIZE, Constants.WORLD_SIZE);
28
29      // TODO: Initialize the ShapeRenderer
30      renderer = new ShapeRenderer();
31
32      // TODO: Set autoShapeType(true) on the ShapeRenderer
33      renderer.setAutoShapeType(true);
34
35      // TODO: Create a new Icicle in the middle of the world
36      icicle = new Icicle(new Vector2(x: Constants.WORLD_SIZE / 2, y: Constants.WORLD_SIZE / 2));
37
38  }
39
40  @Override
41  public void resize(int width, int height) {
42      // TODO: Ensure that the viewport updates correctly
43      iciclesViewport.update(width, height, centerCamera: true);
44  }
```


Classe ICicleScreen.java

```
50      @Override
51      public void render(float delta) {
52
53          // TODO: Apply the viewport
54          iciclesViewport.apply( centerCamera: true);
55
56          // TODO: Clear the screen to the background color
57          Gdx.gl.glClearColor(Constants.BACKGROUND_COLOR.r,
58                               Constants.BACKGROUND_COLOR.g,
59                               Constants.BACKGROUND_COLOR.b,
60                               alpha: 1);
61          Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);
62
63          // TODO: Set the ShapeRenderer's projection matrix
64          renderer.setProjectionMatrix(iciclesViewport.getCamera().combined);
65
66          // TODO: Draw the Icycle
67
68          renderer.begin(ShapeType.Filled);
69          icicle.render(renderer);
70          renderer.end();
71      }
```


Classe ICicleScreen.java

- Embora tenhamos que implementar todos os métodos da interface Screen, nem todos tem utilidade no momento.

```
73
74 
75
76
77
78
79 
80
81
82
83
84
85 
86
87
88 }
```

```
@Override
public void pause() {

}

@Override
public void resume() {

}

// TODO: Dispose of the ShapeRenderer
@Override
public void hide() {
    renderer.dispose();
}
```

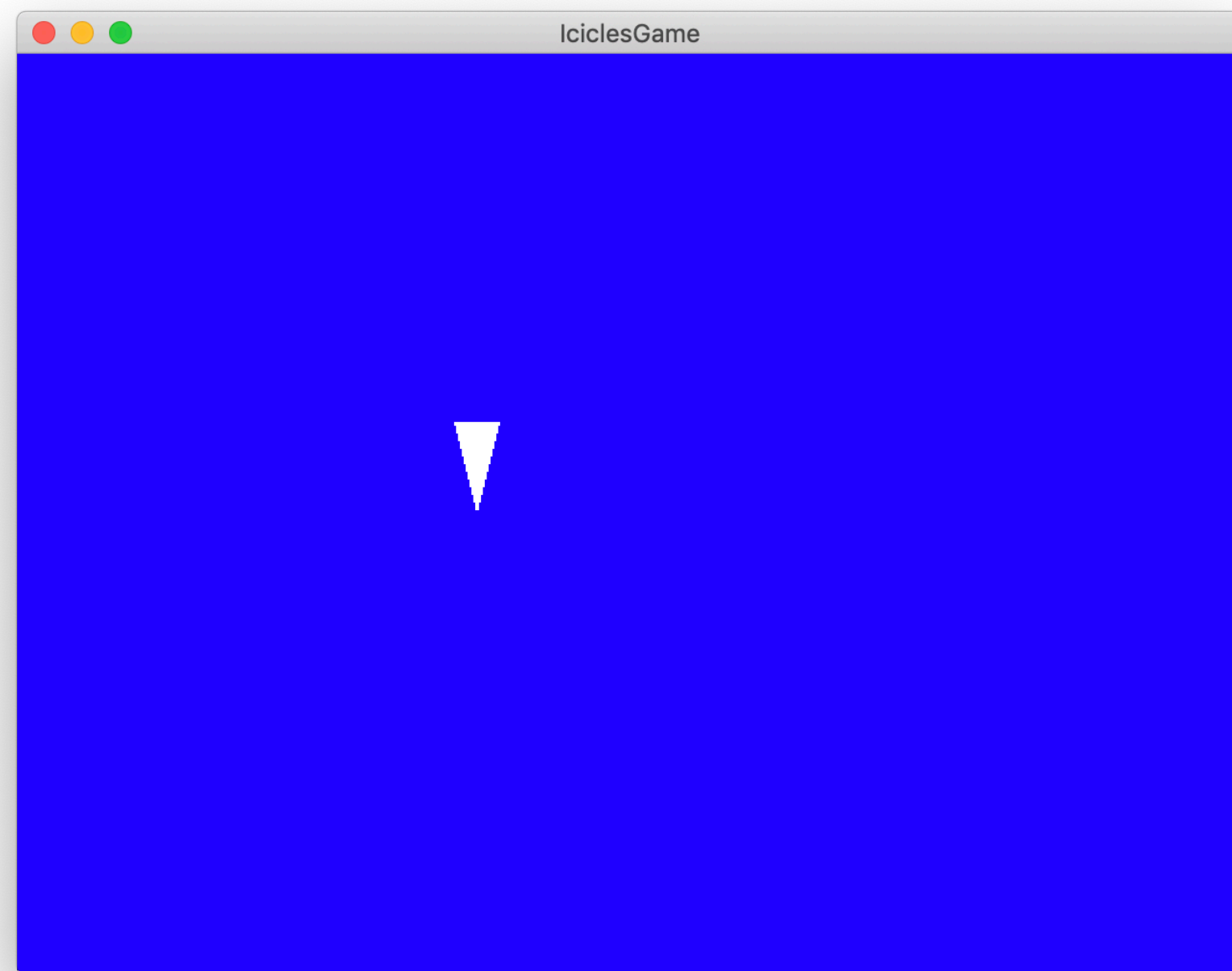

Classe ICicle.java

- Este é o objeto que representa o nosso icicle.

```
3  import com.badlogic.gdx.graphics.glutils.ShapeRenderer;
4  import com.badlogic.gdx.graphics.glutils.ShapeRenderer.ShapeType;
5  import com.badlogic.gdx.math.Vector2;
6
7  public class Icicle {
8
9      public static final String TAG = Icicle.class.getName();
10
11      // TODO: Add a Vector2 position
12      private Vector2 position;
13
14      // TODO: Add a constructor that sets the position
15      public Icicle(Vector2 position) { this.position = position; }
```


Classe ICicle.java

- E definimos no ShapeRenderer no método render qual a cor, preenchimento e desenhar o objeto através do método triangle.



```
19 // TODO: Add a render function that takes a ShapeRenderer
20 @
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
// TODO: Add a render function that takes a ShapeRenderer
public void render(ShapeRenderer renderer) {

    // TODO: Set the ShapeRenderer's color
    renderer.setColor(Constants.ICICLE_COLOR);

    // TODO: Set the ShapeType
    renderer.set(ShapeType.Filled);

    // TODO: Draw the icicle using the size constants
    renderer.triangle(
        position.x,
        position.y,
        x2: position.x - Constants.ICICLES_WIDTH / 2,
        y2: position.y + Constants.ICICLES_HEIGHT,
        x3: position.x + Constants.ICICLES_WIDTH / 2,
        y3: position.y + Constants.ICICLES_HEIGHT
    );
}
```