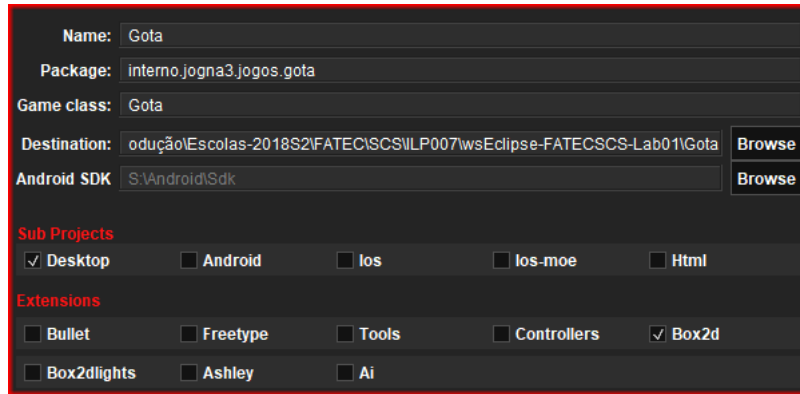


Jogo Básico LibGDX Parte I (baseado em A simple game - <https://libgdx.com/dev/simple-game/>)

Neste tutorial serão explorados vários componentes construtivos de um jogo: acesso básico a arquivos, limpeza de tela, desenho de imagens, uso da câmera, processamento básico de entrada e o uso de efeitos de som.

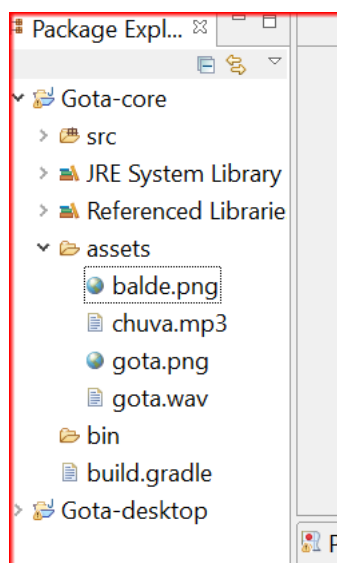
1. Configurar o jogo:



2. Baixar os recursos ("assets") (alguns é preciso criar uma conta para baixar:

- Som de gota d'água de jungle (renomear para gota.wav): <http://www.freesound.org/people/jungle/sounds/30341/>
- Som de chuva de acclivity (renomear para chuva.mp3): <http://www.freesound.org/people/acclivity/sounds/28283/>
- Imagem de gota d'água de mvdv (renomear para gota.png): <https://www.box.com/s/pegrdkwj16guhpm48nit>
- Imagem de balde de mvdv (: <https://www.box.com/s/605bvd1wuqubtutbyf4x>

3. Copiar os recursos para o diretório assets e excluir badlogic.jpg:



4. Configurar a classe de início (DesktopLauncher.java):

Nome da aplicação: Gota;
Dimensões: 800 X 480;

```
public static void main (String[] arg) {  
    LwjglApplicationConfiguration config = new LwjglApplicationConfiguration();  
    config.title = "Gota";  
    config.width = 800;  
    config.height = 480;  
    new LwjglApplication(new Gota(), config);  
}
```

5. Código

5.1 – Carregamento dos recursos (assets):

Consiste em criar referências a eles. De maneira geral isso é feito no método `ApplicationAdapter.create()`.

```
package interno.jogna3.jogos.gota;  
  
import com.badlogic.gdx.ApplicationAdapter;  
import com.badlogic.gdx.Gdx;  
import com.badlogic.gdx.audio.Music;  
import com.badlogic.gdx.audio.Sound;  
import com.badlogic.gdx.graphics.Texture;  
  
public class Gota extends ApplicationAdapter {  
    private Texture gotaImagem;  
    private Texture baldeImagem;  
    private Sound gotaSom;  
    private Music chuvaMusica;  
  
    @Override  
    public void create() {  
        // carrega a imagem para a gota e o balde, ambos de 64X64 pixels  
        gotaImagem = new Texture(Gdx.files.internal("gota.png"));  
        baldeImagem = new Texture(Gdx.files.internal("balde.png"));  
  
        // carrega o efeito sonoro gota e a "música de fundo chuva"  
        gotaSom = Gdx.audio.newSound(Gdx.files.internal("gota.wav"));  
        chuvaMusica = Gdx.audio.newMusic(Gdx.files.internal("chuva.mp3"));  
  
        // inicia a execução da música de fundo imediatamente  
        chuvaMusica.setLooping(true);  
        chuvaMusica.play();  
  
        // ... tem mais código aqui para completar o método...  
    }  
  
    // o resto da classe foi omitida para deixar mais claro o ponto
```

Assim, cada recurso tem um campo de referência. As texturas são as imagens carregadas através de um manipulador de arquivo (`Filehandle`) obtido através do método `Gdx.files`. O tipo *internal* de refere a diretório `assets` do projeto. Efeitos sonoros são armazenados na memória e música é transmitida (*streamed*) de onde ela estiver localizada. Como regra informal considerar som as peças com menos de 10 segundos e música peças mais longas (são carregados via `Filehandle` através dos métodos `Gdx.audio.newSound()` e `Gdx.audio.newMusic()` respectivamente). As últimas duas instruções colocam a música em *looping* e iniciam a execução.

5.2 – Adicionando uma câmera (Camera) e um sprite (SpriteBatch)

Camera garante que a resolução será 800x480 pixels qualquer que seja a resolução da tela e SpriteBatch é uma classe especial para desenhar imagens 2D tais como as texturas que foram carregadas.

Adicionar os campos:

```
private OrthographicCamera camera;  
private SpriteBatch batch;
```

No método create() criar a câmera:

```
camera = new OrthographicCamera();  
camera.setToOrtho(false, 800, 480);
```

Que assegura que a câmera sempre mostrará uma área de 800X480 do mundo do jogo como uma janela virtual.

... e o SpriteBatch:

```
batch = new SpriteBatch();
```

5.3 – Adicionando o balde

Balde e gotas tem uma posição x e y, uma largura, uma altura e uma representação gráfica (textura) no plano 800X480. A classe Rectangle pode ser utilizada para armazenar essas posições e tamanhos.

Assim, adicionar a biblioteca e o campo:

```
import com.badlogic.gdx.math.Rectangle;  
  
private Rectangle balde;
```

Instanciar o retângulo no método create():

```
// cria um retângulo que representa logicamente o balse  
balde = new Rectangle();  
balde.x = 800 / 2 - 64 / 2; // centraliza o balde na horizontal  
balde.y = 20; // o canto esquerdo inferior do balde tem que estar  
// 20 pixels acima da borda inferior da tela  
balde.width = 64;  
balde.height = 64;
```

Para renderizar o balde no método render():

```
public void render() {  
    // limpa a tela com azul escuro. Os argumentos para glClearColor são  
    // os componentes RGB (vermelho, verde e azul) e o Canal alfa com  
    // faixa de variação [0,1] da cor utilizada para limpar a tela  
    Gdx.gl.glClearColor(0, 0, 0.2f, 1);  
    Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);  
  
    // diga para a câmera que atualize suas matrizes.  
    camera.update();  
  
    // diga ao SpriteBatch que renderize no sistema  
    // de coordenadas especificado pela câmera.  
    batch.setProjectionMatrix(camera.combined);
```

```
// iniciar um batch (lote) e desenhe o balde
batch.begin();
batch.draw(baldeImagem, balde.x, balde.y);
batch.end();
```

Código até aqui:

```
package interno.jogna3.jogos.gota;

import com.badlogic.gdx.ApplicationAdapter;
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.audio.Music;
import com.badlogic.gdx.audio.Sound;
import com.badlogic.gdx.graphics.GL20;
import com.badlogic.gdx.graphics.OrthographicCamera;
import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.graphics.g2d.SpriteBatch;
import com.badlogic.gdx.math.Rectangle;

public class Gota extends ApplicationAdapter {
    private Texture gotaImagem;
    private Texture baldeImagem;
    private Sound gotaSom;
    private Music chuvaMusica;
    private OrthographicCamera camera;
    private SpriteBatch batch;
    private Rectangle balde;

    @Override
    public void create() {
        // carrega a imagem para a gota e o balde, ambos de 64x64 pixels
        gotaImagem = new Texture(Gdx.files.internal("gota.png"));
        baldeImagem = new Texture(Gdx.files.internal("balde.png"));

        // carrega o efeito sonoro gota e a "música de fundo chuva
        gotaSom = Gdx.audio.newSound(Gdx.files.internal("gota.wav"));
        chuvaMusica = Gdx.audio.newMusic(Gdx.files.internal("chuva.mp3"));

        // inicia a execução da música de fundo imediatamente
        chuvaMusica.setLooping(true);
        chuvaMusica.play();

        // cria a camera e o SpriteBatch
        camera = new OrthographicCamera();
        camera.setToOrtho(false, 800, 480);
        batch = new SpriteBatch();

        // cria um retângulo que representa logicamente o balse
        balde = new Rectangle();
        balde.x = 800 / 2 - 64 / 2; // centraliza o balde na horizontal
        balde.y = 20; // o canto esquerdo inferior do balde tem que estar
        // 20 pixels acima da borda inferior da tela
        balde.width = 64;
        balde.height = 64;

        // ... tem mais código aqui para completar o método...
    }

    public void render() {
        // limpa a tela com azul escuro. o argumento para glClearColor são
        // os componentes RGB (vermelho, verde e azul) e Canal alfa com
        // faixa de variação [0,1] da cor utilizada para limpar a tela
        Gdx.gl.glClearColor(0, 0, 0.2f, 1);
        Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);

        // diga para a câmera que atualize suas matrizes.
        camera.update();

        // diga ao SpriteBatch que renderize no sistema
        // de coordenadas especificado pela câmera.
        batch.setProjectionMatrix(camera.combined);

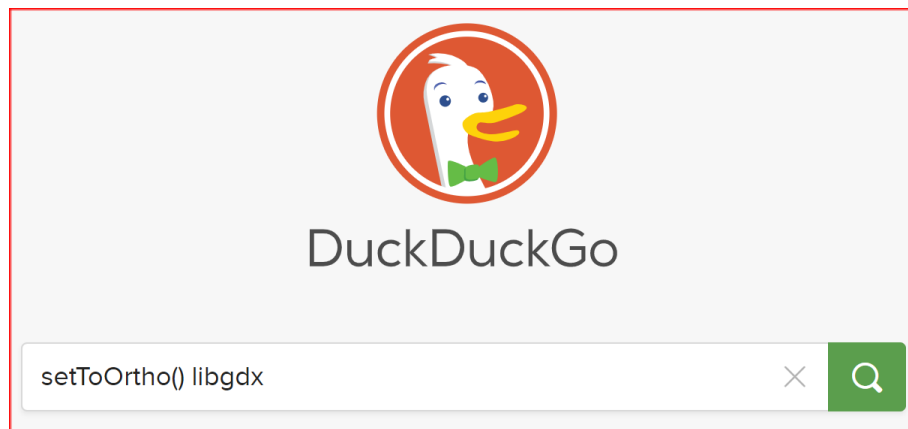
        // inicie um batch (lote) e desenhe o balde
        batch.begin();
        batch.draw(baldeImagem, balde.x, balde.y);
        batch.end();

        // ... tem mais código aqui para completar o método...
    }

    // o resto da classe foi omitida para deixar mais claro o ponto
```

Atividade ILJ003-Framework02 LibGDX Jogo Básico Parte 1

Comentar os recursos disponíveis no LibGDX que foram abordados na primeira parte do jogo de exemplo "Gota". Para isso, procurar no Google ou no Duckduckgo os métodos que foram usados. Exemplo:



OVERVIEW PACKAGE **CLASS** USE TREE DEPRECATED INDEX HELP

PREV CLASS **NEXT CLASS** FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | [FIELD](#) | [CONSTR](#) | [METHOD](#) DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

com.badlogic.gdx.graphics

Class OrthographicCamera

java.lang.Object
[com.badlogic.gdx.graphics.Camera](#)
[com.badlogic.gdx.graphics.OrthographicCamera](#)

public class **OrthographicCamera**
extends [Camera](#)

A camera with orthographic projection.

Author:
mzechner

Field Summary

Fields

Modifier and Type	Field and Description
float	zoom the zoom of the camera

Fields inherited from class [com.badlogic.gdx.graphics.Camera](#)

[combined](#), [direction](#), [far](#), [frustum](#), [invProjectionView](#), [near](#), [position](#), [projection](#), [up](#), [view](#), [viewportHeight](#), [viewportWidth](#)

Constructor Summary

Constructors

Constructor and Description
OrthographicCamera()
OrthographicCamera(float viewportWidth, float viewportHeight) Constructs a new OrthographicCamera, using the given viewport width and height.

Method Summary

All Methods **Instance Methods** **Concrete Methods**

Modifier and Type	Method and Description
void	rotate(float angle) Rotates the camera by the given angle around the direction vector.
void	setToOrtho(boolean yDown) Sets this camera to an orthographic projection using a viewport fitting the screen resolution, centered at (Gdx.graphics.getWidth()/2, Gdx.graphics.getHeight()/2), with the y-axis pointing up or down.
void	setToOrtho(boolean yDown, float viewportWidth, float viewportHeight) Sets this camera to an orthographic projection, centered at (viewportWidth/2, viewportHeight/2), with the y-axis pointing up or down.
void	translate(float x, float y) Moves the camera by the given amount on each axis.
void	translate(Vector2 vec) Moves the camera by the given vector.
void	update() Recalculates the projection and view matrix of this camera and the Frustum planes.
void	update(boolean updateFrustum) Recalculates the projection and view matrix of this camera and the Frustum planes if updateFrustum is true.

Versão utilizada no jogo básico.