Leibniz Universität Hannover

# Handy-Programmierung unter Android

## Niedersächsische Talente-Akademie 2016

Human-Computer Interaction Group

Prof. Dr. Michael Rohs
michael.rohs@hci.uni-hannover.de

# libGDX Documentation

- Home
  - http://libgdx.badlogicgames.com
- Wiki
  - https://github.com/libgdx/libgdx/wiki
- API
  - http://libgdx.badlogicgames.com/nightlies/docs/api/

Leibniz Universität Hannover

# JAVA: VARIABLES, CLASSES, OBJECTS

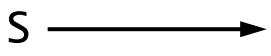
# Variables, Classes, Objects

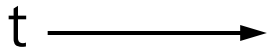
```
public class MyGame {
    Sound s; // variable name s, class Sound, no sound object yet
    Sound t; // variable name t, class Sound, no sound object yet
    public void create() {
        // create Sound object from file, give it name s
        s = Gdx.audio.newSound(Gdx.files.internal("explode.wav"));
        // give this sound a second name: t
        t = s;
        ...
        // create Sound object from file, give it name t
        t = Gdx.audio.newSound(Gdx.files.internal("boing.wav"));
        ...
    }
}
```

# Variables, Classes, Objects (step 1)


```
public class MyGame {
    Sound s; // (step 1)
    Sound t; // (step 2)
    public void create() {
        // create Sound object from file, give it name s
        s = Gdx.audio.newSound(Gdx.files.internal("explode.wav")); // (step 3)
        // give this sound a second name: t
        t = s; // (step 4)
        ...
        // create Sound object from file, give it name t
        t = Gdx.audio.newSound(Gdx.files.internal("boing.wav")); // (step 5)
        ...
    }
}
```

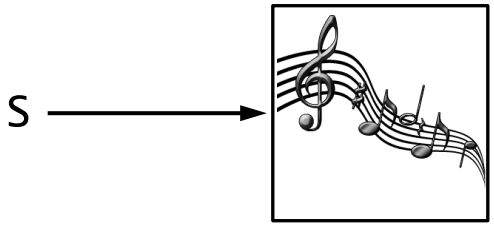
# Variables, Classes, Objects (step 2)



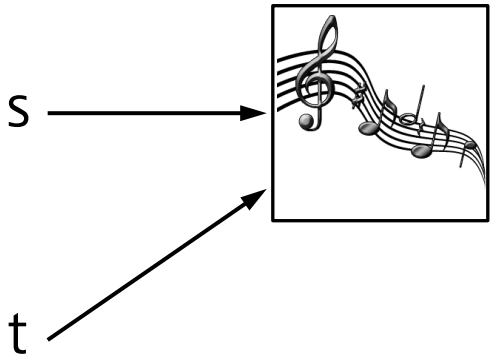

```
public class MyGame {
    Sound s; // (step 1)
    Sound t; // (step 2)
    public void create() {
        // create Sound object from file, give it name s
        s = Gdx.audio.newSound(Gdx.files.internal("explode.wav")); // (step 3)
        // give this sound a second name: t
        t = s; // (step 4)
        ...
        // create Sound object from file, give it name t
        t = Gdx.audio.newSound(Gdx.files.internal("boing.wav")); // (step 5)
        ...
    }
}
```

# Variables, Classes, Objects (step 3)



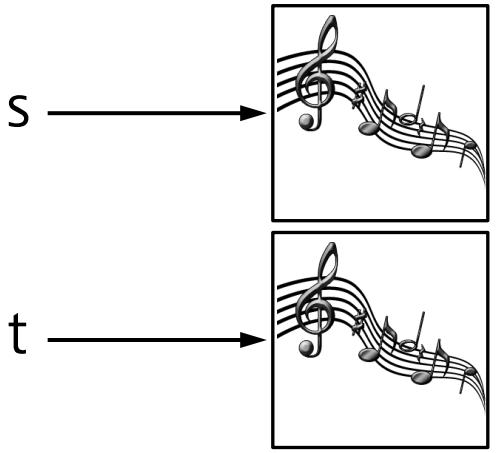

```
public class MyGame {
    Sound s; // (step 1)
    Sound t; // (step 2)
    public void create() {
        // create Sound object from file, give it name s
        s = Gdx.audio.newSound(Gdx.files.internal("explode.wav")); // (step 3)
        // give this sound a second name: t
        t = s; // (step 4)
        ...
        // create Sound object from file, give it name t
        t = Gdx.audio.newSound(Gdx.files.internal("boing.wav")); // (step 5)
        ...
    }
}
```

# Variables, Classes, Objects (step 4)




```
public class MyGame {
    Sound s; // (step 1)
    Sound t; // (step 2)
    public void create() {
        // create Sound object from file, give it name s
        s = Gdx.audio.newSound(Gdx.files.internal("explode.wav")); // (step 3)
        // give this sound a second name: t
        t = s; // (step 4)
        ...
        // create Sound object from file, give it name t
        t = Gdx.audio.newSound(Gdx.files.internal("boing.wav")); // (step 5)
        ...
    }
}
```

# Variables, Classes, Objects (step 5)




```
public class MyGame {
    Sound s; // (step 1)
    Sound t; // (step 2)
    public void create() {
        // create Sound object from file, give it name s
        s = Gdx.audio.newSound(Gdx.files.internal("explode.wav")); // (step 3)
        // give this sound a second name: t
        t = s; // (step 4)
        ...
        // create Sound object from file, give it name t
        t = Gdx.audio.newSound(Gdx.files.internal("boing.wav")); // (step 5)
        ...
    }
}
```

# COORDINATE SYSTEMS

# Rendering Shapes

- Shapes: Circles, ellipses, rectangles, polygons, lines, polylines, etc.
- Class variable

```
ShapeRenderer sr;
```

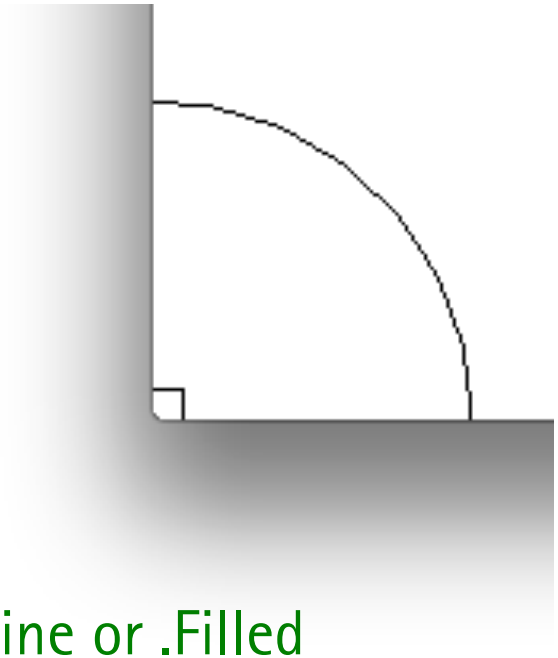
- Initialization

```
public void create() {
    sr = new ShapeRenderer(); ...
}
```

- Update

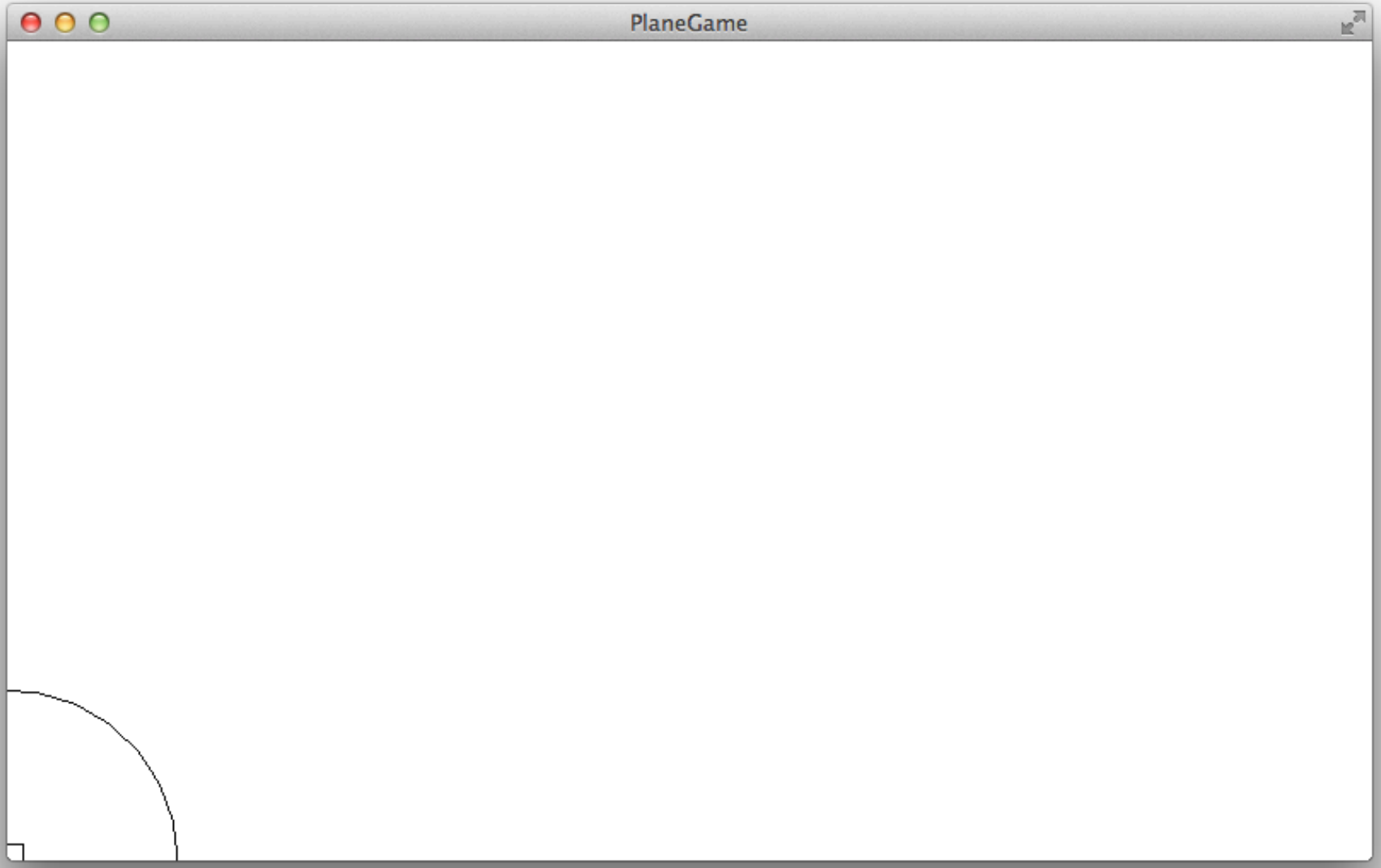
```
 private void updateWorld() {
      float deltaTime = Gdx.graphics.getDeltaTime();
      angle += deltaTime * 360; // rotate 360° per second
 }
```

# Rendering Shapes

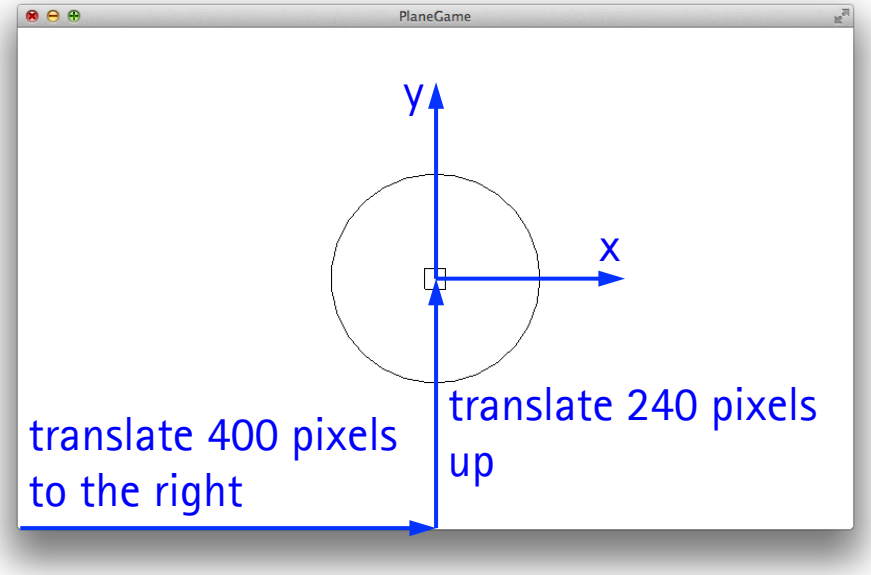


- Drawing

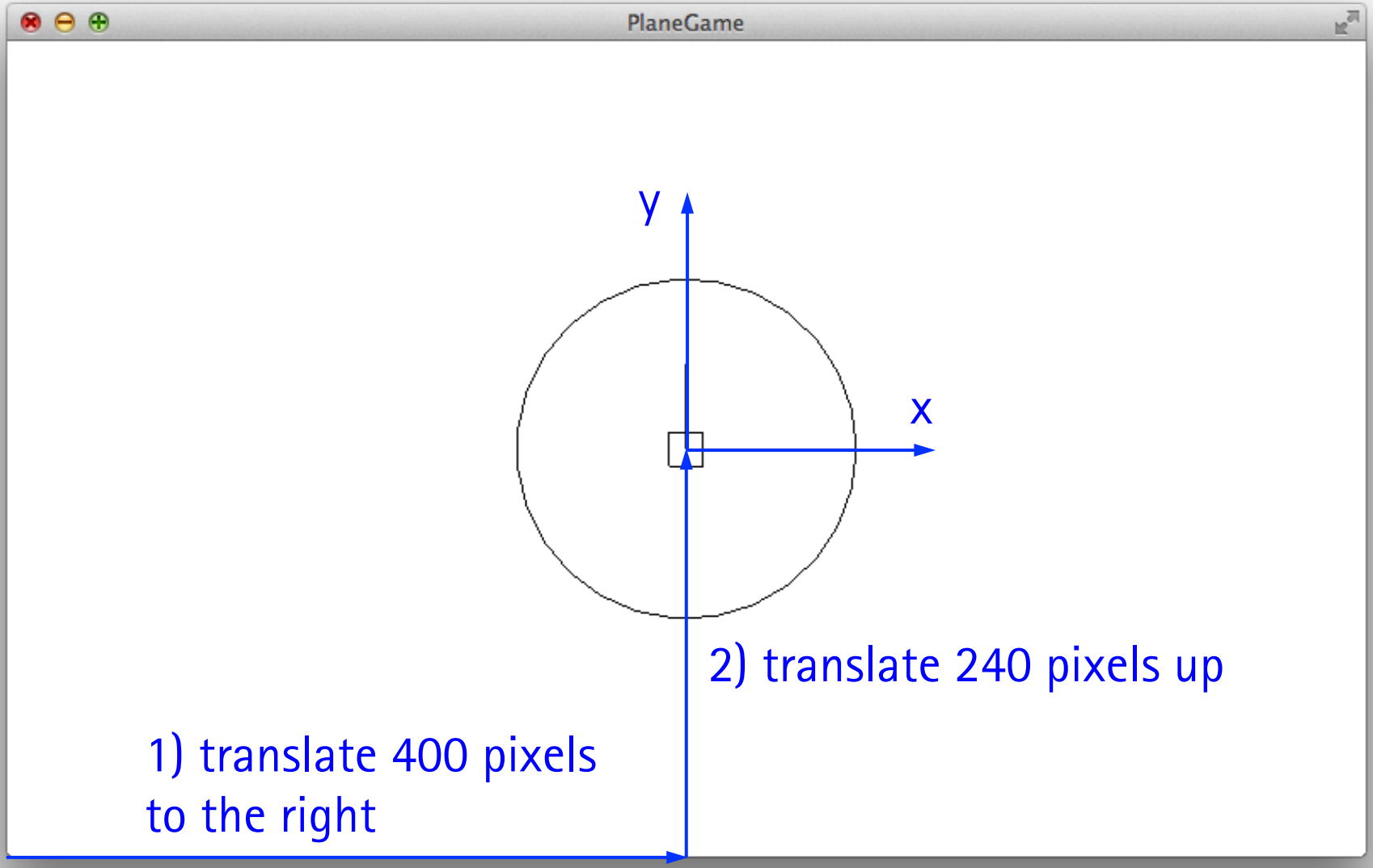
```
private void drawWorld() {
    sr.begin(ShapeType.Line); // .Line or .Filled
    sr.setColor(0, 0, 0, 1); // red, gree, blue, alpha; float; 0..1
                             // alpha: 0 = transparent, 1 = opaque
    sr.circle(0, 0, 100); // x, y, radius
    sr.rect(-10, -10, 20, 20); // x, y, width, height
    sr.end();
}
```


PlaneGame

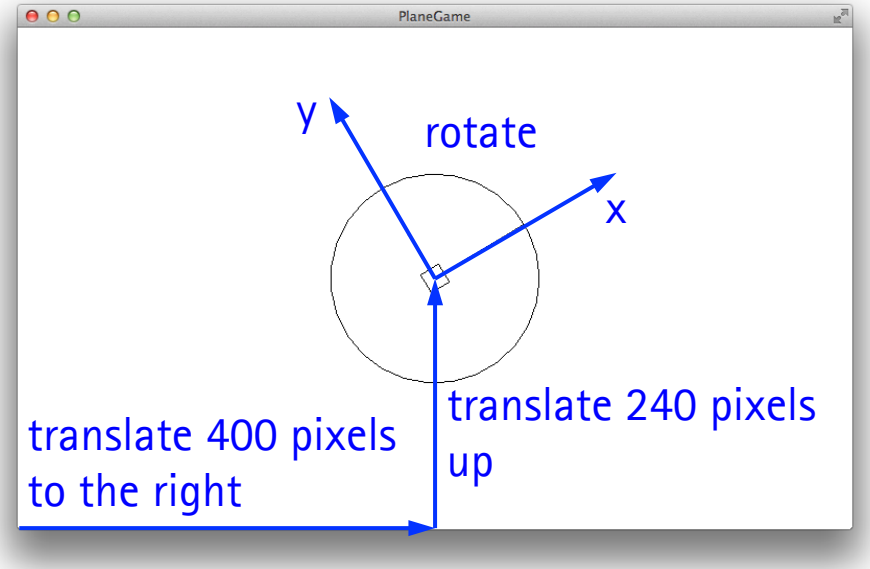
# Translate
# Coordinate System



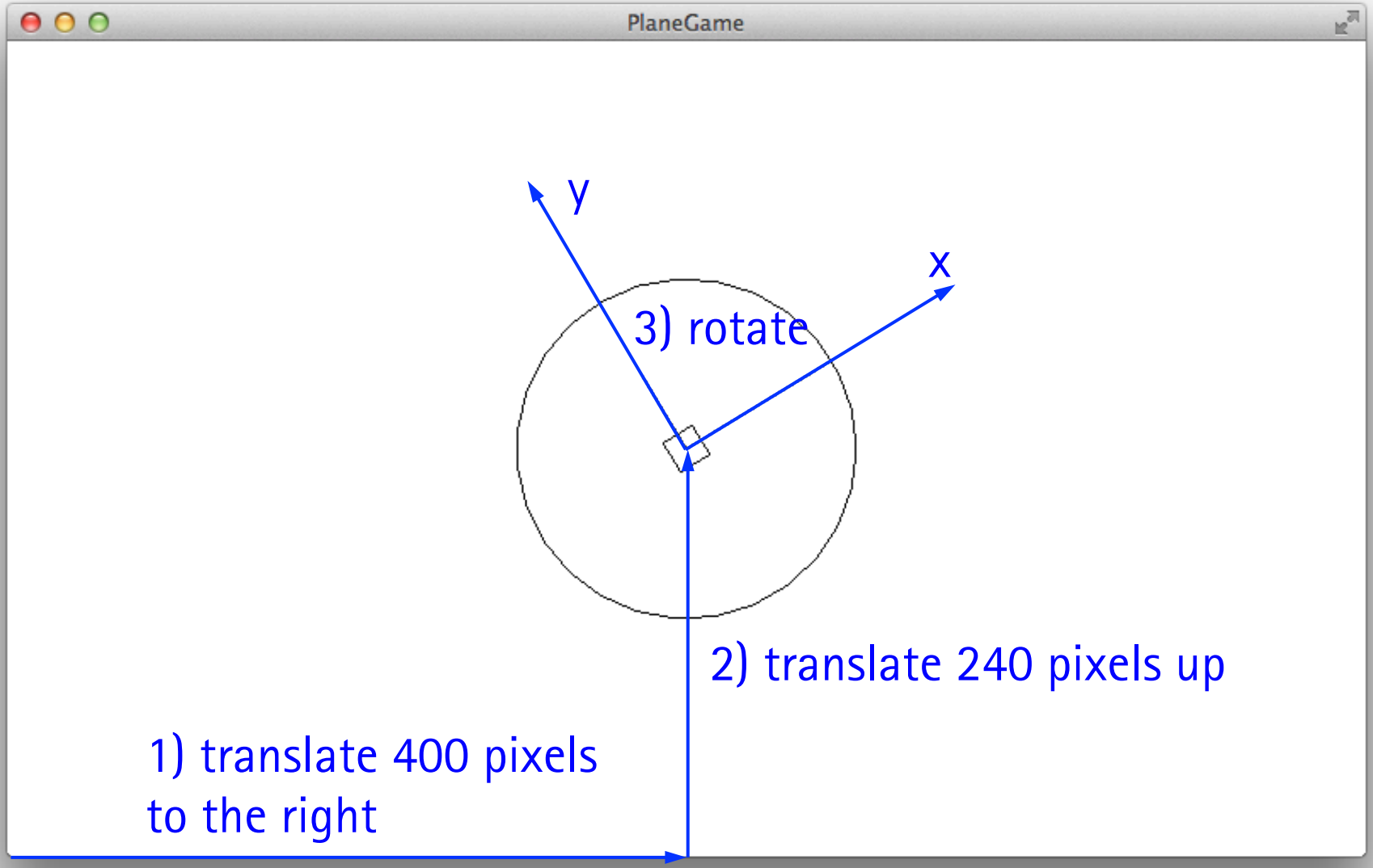

```
Matrix4 m = sr.getTransformMatrix().cpy();
sr.begin(ShapeType.Line); // .Line or .Filled
sr.setColor(0, 0, 0, 1); // red, gree, blue, alpha

sr.translate(400, 240, 0); // translate coordinate system 400 right, 240 up

sr.circle(0, 0, 100); // x, y, radius
sr.line(0, 0, 100, 0); // x1, y1, x2, y2
sr.line(0, 0, 0, 50); // x1, y1, x2, y2
sr.rect(-10, -10, 20, 20); // x, y, width, height
sr.end();
sr.setTransformMatrix(m); // reset coordinate system
```


PlaneGame
y
x
2) translate 240 pixels up
1) translate 400 pixels
to the right

# Translate and Rotate Coordinate System

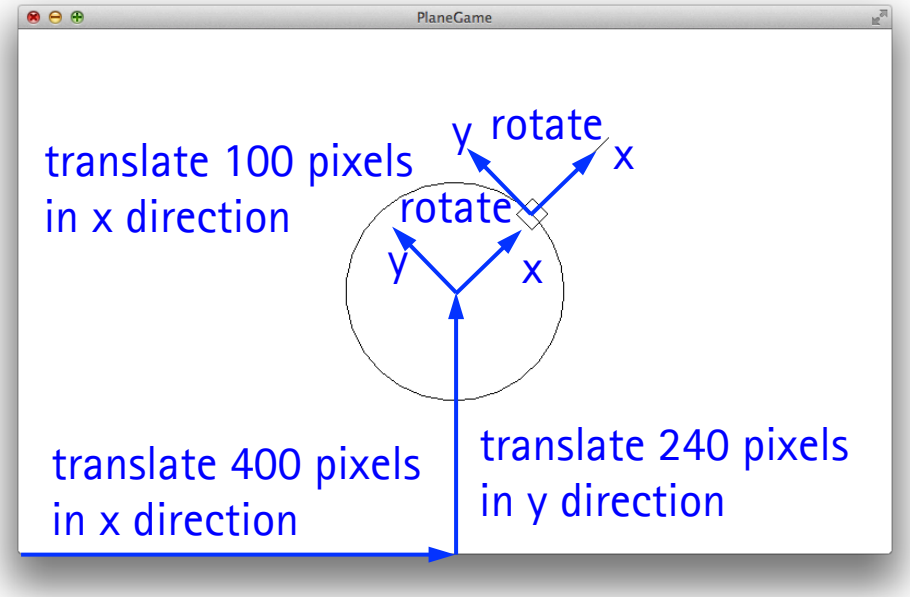


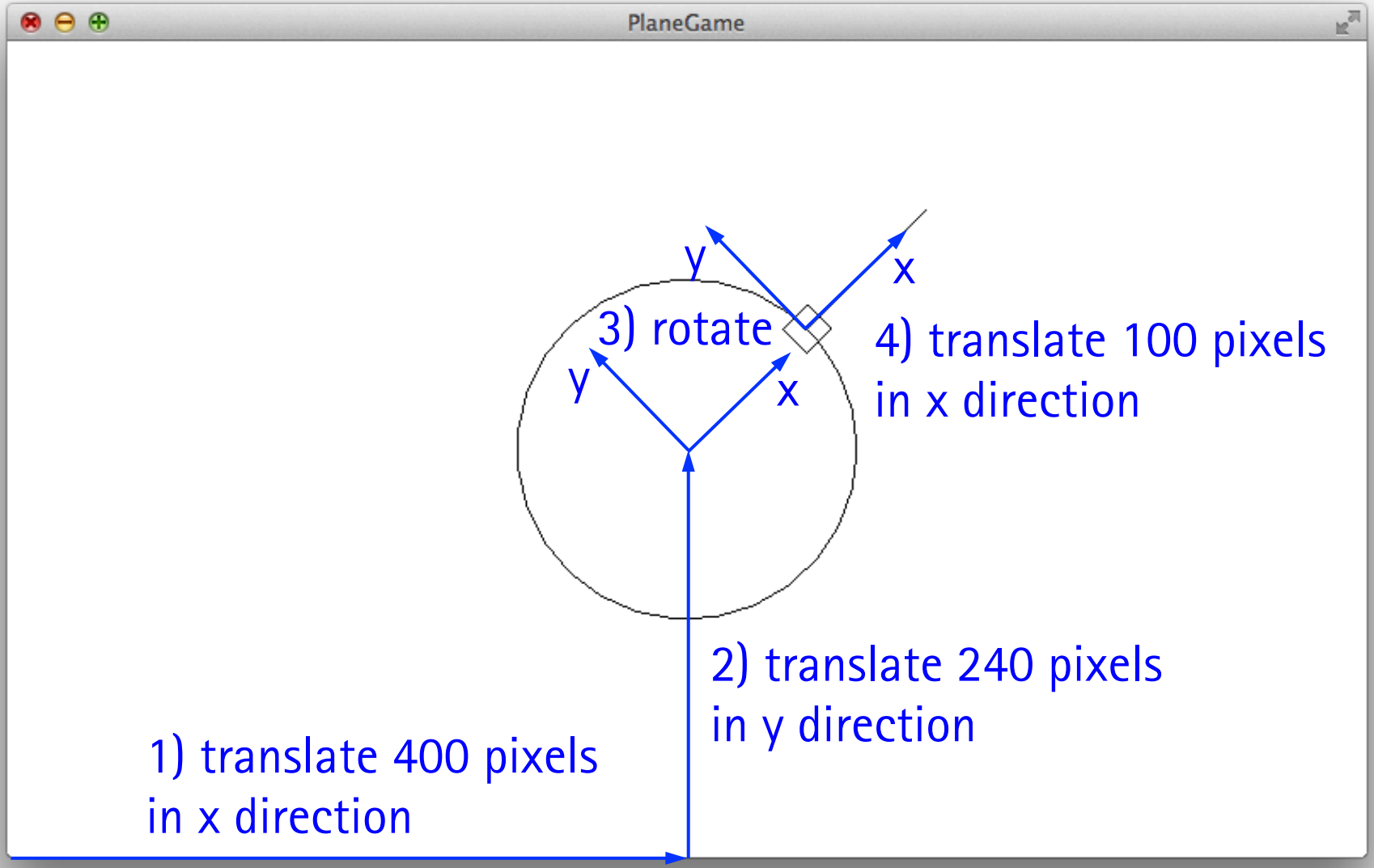

```
Matrix4 m = sr.getTransformMatrix().cpy();
sr.begin(ShapeType.Line); // .Line or .Filled
sr.setColor(0, 0, 0, 1); // red, gree, blue, alpha

sr.translate(400, 240, 0); // translate coordinate system 400 right, 240 up
sr.circle(0, 0, 100); // x, y, radius
sr.rotate(0, 0, 1, angle); // rotate around z-axis in degrees
sr.line(0, 0, 100, 0); // x1, y1, x2, y2
sr.line(0, 0, 0, 50); // x1, y1, x2, y2
sr.rect(-10, -10, 20, 20); // x, y, width, height
sr.end();
sr.setTransformMatrix(m); // reset coordinate system
```


PlaneGame
y
x
3) rotate
2) translate 240 pixels up
1) translate 400 pixels
to the right

# Rendering Shapes

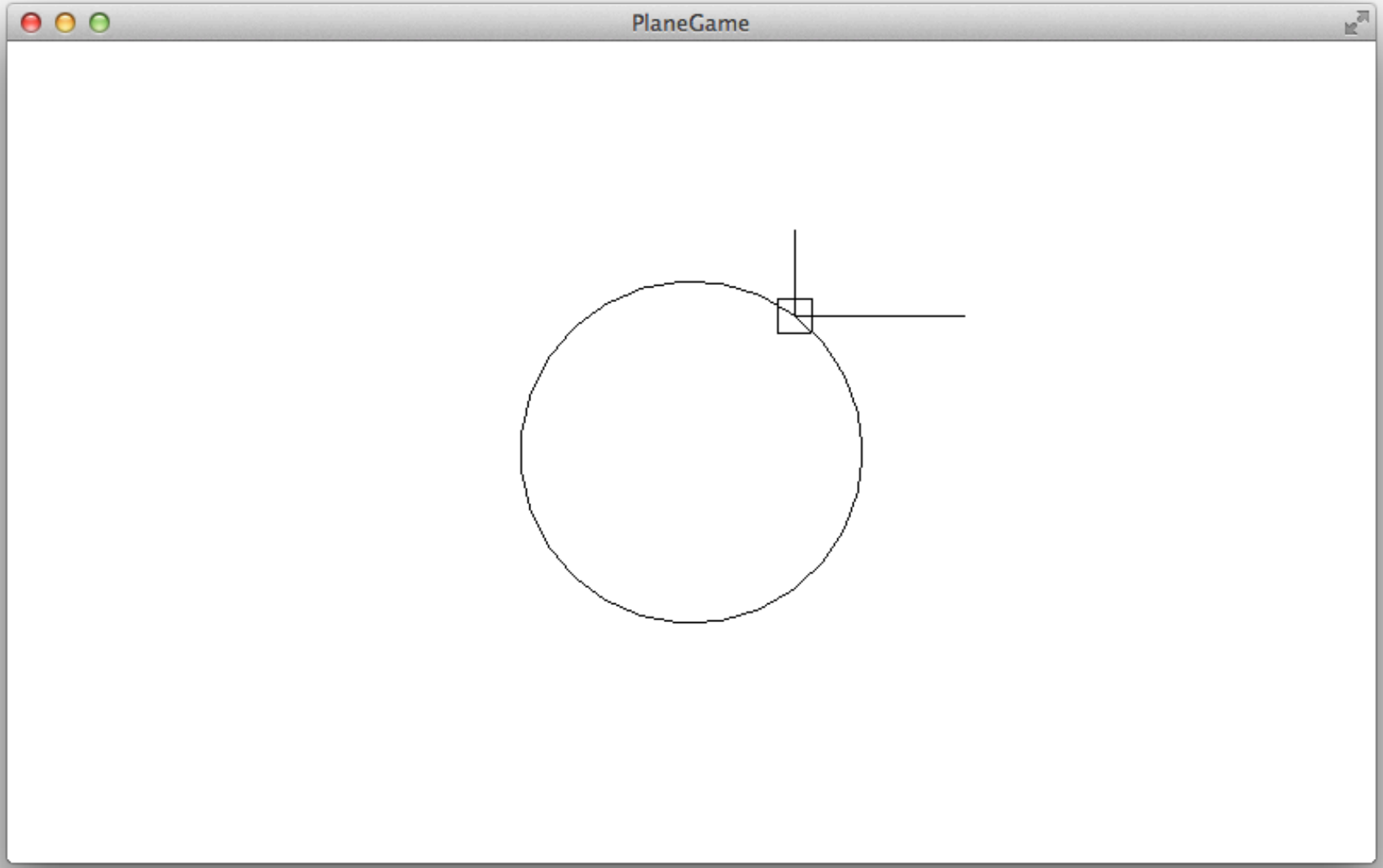
```
Matrix4 m = sr.getTransformMatrix().cpy();
sr.begin(ShapeType.Line); // .Line or .Filled
sr.setColor(0, 0, 0, 1); // rgba
sr.translate(400, 240, 0);
sr.circle(0, 0, 100); // x, y, radius
sr.rotate(0, 0, 1, angle);
sr.translate(100, 000, 0);
sr.line(0, 0, 100, 0); // x1, y1, x2, y2
sr.line(0, 0, 0, 50); // x1, y1, x2, y2
sr.rect(-10, -10, 20, 20); // x, y, width, height
sr.setTransformMatrix(m);
sr.end();
```




PlaneGame
y
x
3) rotate
y
x
4) translate 100 pixels
in x direction
2) translate 240 pixels
in y direction
1) translate 400 pixels
in x direction

# Rendering Shapes

```
Matrix4 m = sr.getTransformMatrix().cpy();
sr.begin(ShapeType.Line); // .Line or .Filled
sr.setColor(0, 0, 0, 1); // red, gree, blue, alpha
sr.translate(400, 240, 0);
sr.circle(0, 0, 100); // x, y, radius
sr.rotate(0, 0, 1, angle); // positive rotation
sr.translate(100, 000, 0);
sr.rotate(0, 0, 1, -angle); // negative rotation, compensate for positive rotation
sr.line(0, 0, 100, 0); // x1, y1, x2, y2
sr.line(0, 0, 0, 50); // x1, y1, x2, y2
sr.rect(-10, -10, 20, 20); // x, y, width, height
sr.setTransformMatrix(m);
sr.end();
```


PlaneGame

Leibniz Universität Hannover

# MULTIPLE SCREENS

# MyGame

```
public class MyGame extends Game {
    ...
    public void create() {
        ...
        setScreen(new MainMenuScreen(this));
    }
    public void render() {
        super.render();
    }
    public void dispose() {...}
}
```

# MainMenuScreen

```
public class MainMenuScreen implements Screen {
    final MyGame game;
    OrthographicCamera camera;
    ...
    public MainMenuScreen(final MyGame game) { ... }

    public void render(float delta) {
        if (Gdx.input.isTouched()) {
            game.setScreen(new GameScreen(game));
            dispose();
        }
        ...
    }
```

# MainMenuScreen

```
    public void resize(int width, int height) { ... }
    public void show() { ... }
    public void hide() { ... }
    public void pause() { ... }
    public void resume() { ... }
    public void dispose() { ... }
}
```

# GameScreen

```
public class GameScreen implements Screen {
    final MyGame game;
    OrthographicCamera camera;
    ...
    public GameScreen(final MyGame game) { ... }
    public void render(float delta) { ... }
    public void resize(int width, int height) { ... }
    public void show() { ... }
    public void hide() { ... }
    public void pause() { ... }
    public void resume() { ... }
    public void dispose() { ... }
}
```