CS 402: Mobile Development

Michael Ziray - michaelziray@boisestate.edu

Android Game Development

Additional Reading

Android Game Tutorials

The ABCs of Android game development

Native Android Development

Games

Games are hard. We need good:

- Story (beginning, middle, end)
- Gameplay
- Game flow
- Replayability

Use this as a basic checklist for your game.

Story

Every game has some kind of story or theme.

Start with the story. It’ll define the mood, genre, and interaction.

Stories don’t have to be involved, but needs to imerse the player in that world.

Story

It certainly needs an end. Why is the player playing?

Certainly not because the gameplay is fun. Define an objective.

End to a level, and end to a game.

Game Play

Introduce the player to the game.

Introduce them to the controls.

Get them to play more. Always have a call to action to play the next thing.

Game Flow

The game flow is how the game proceeds through it’s story and the player through the levels or scenes.

Good game flow doesn’t jar the player out of the environment or frustrate them without purpose. It proceeds seamlessly from one step to another.

Replayability

You want the player to be able to play the game over and over, especially if all levels share the same type of game play. (ex: Mario)

Components of a Game

Game loop

Game states (playing, paused, scene, loading, etc)

Game logic

Physics

Collisions

User Input

Artificial Intelligence

Game Loop

Infinite but breakable loop that continuously evaluates, reacts and draws the game world.

Works with game state machine to determine how to function.

Game State Machine

Has various states to determine which part of the game you're in.

Implemented using subclasses of a State class with overridden methods to control basic functionality.

Example

```
public class BaseStateMachine{
  public void gameLoop(){
    // empty
  }
  public void onPause(){
    // empty
  }
}
```

Example

```
public class MenuState extends BaseStateMachine{
  public void gameLoop(){
    showMenu();
    registerListeners();
  }
  public void onPause(){
    hideMenu();
    unregisterListeners();
  }
}
```

Example

```
public class PlayState extends BaseStateMachine{
  public void gameLoop(){
    drawScene();
    registerListeners();
  }
  public void onPause(){
    showPauseMenu();
    unregisterListeners();
  }
}
```

Example

```
private void mainGameLoop(){
  while( this.isPlaying ){
    currentState.gameLoop();

    switch( this.curentUserInput ){
      case KEY_ESC: this.currentState = new MenuState(); break;
      case KEY_SPACE: this.currentState = new PlayState(); break;
  }
}
```

Game Logic

Can become spaghetti code very quickly without proper software engineering patterns.

Engineering a Game

Singletons:

Health, ammo, loot, experience, etc can be stored into a player singleton. Can also have one for Game values such as difficulty and music settings.

```
Player.ammo += 50;
Player.health -= 10;
```

Engineering a Game

Factory:

Levels or enemies can be loaded via factories

```
LevelFactory.loadLevel(“Level of doom”,
  Game.playerDifficulty);


EnemyFactory.spawnEnemy(ENEMY_TYPE_ORC, {30, 100, 60},
{health: 300, level: 32}); // TYPE, LOCATION, ATTRIBUTES
```

Engineering a Game

Use your language and patterns!!

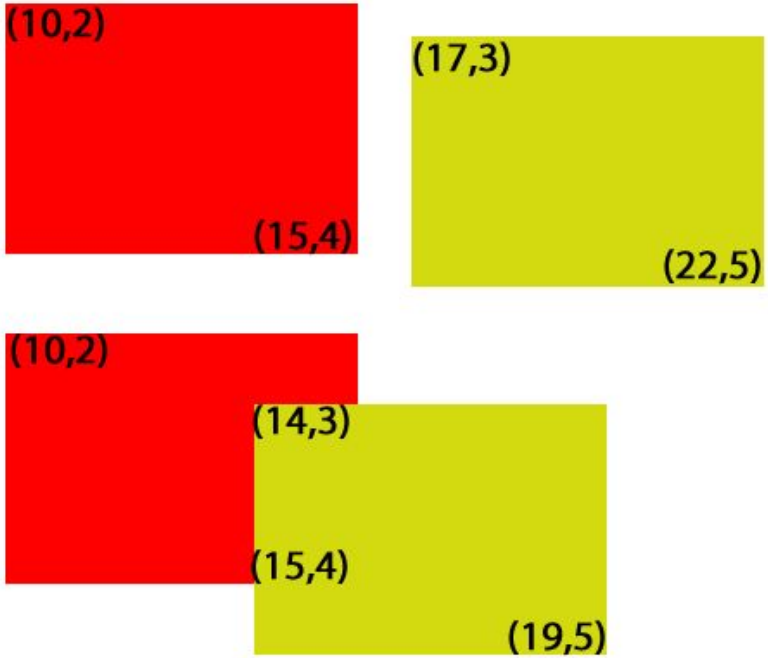
There is almost nothing more complex than a game’s logic.

Physics

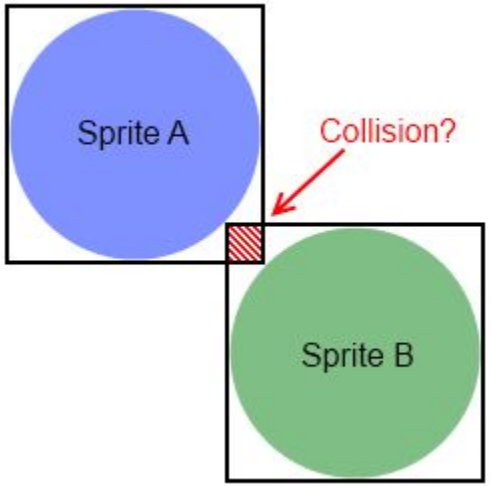
Almost all games have physics to some degree.

Use a 3rd party physics engine for ease of development. It’s also already tested.

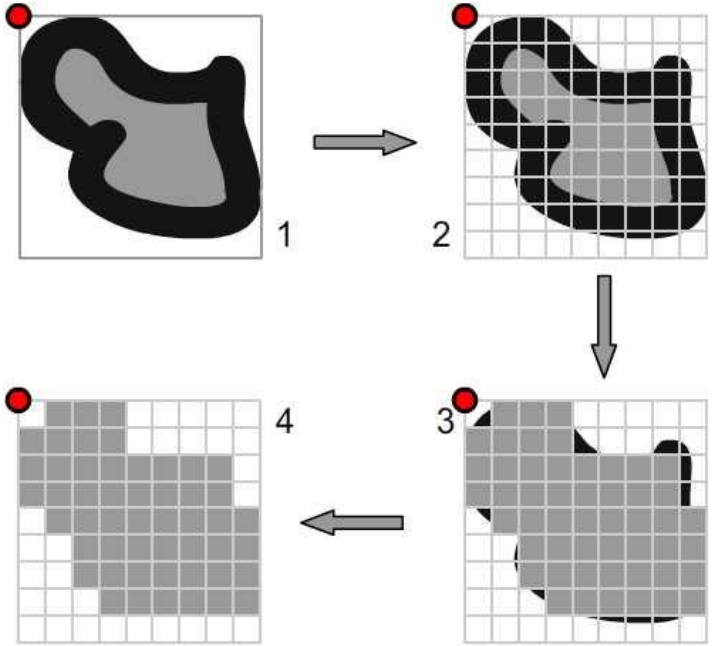
Collisions



Collisions



Collisions



User Input

On mobile, we have:

Tap

Long tap

Slide

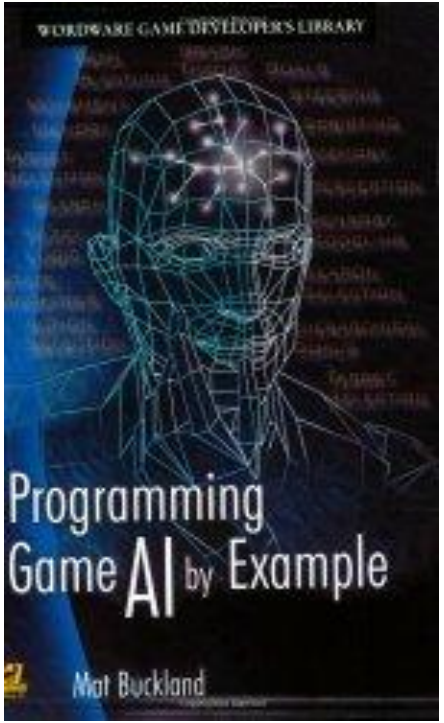
Gestures (pinch, pull, shapes, etc)

Orientation

Hardware buttons

Artificial Intelligence

Programming Game AI by Example



3D Game Terminology

Meshes: Your 3D object's geometry

Material: Skin or texture of your mesh

Shader: Algorithm to take an incoming photon and how that reflects, scatters or gets absorbed.

3D Game Terminology

Unity Prefab: Combination of meshes, materials, shaders, animations, scripts etc to allow the developer to drag and drop these prefabricated objects into the scene and have them just work.

Unreal Blueprints

Development SDKS

Pros and Cons

Lib GDX

Pros:

Written in Java

Windows, Mac, Linux, Android, iOS and HTML5

Low level

Heavy OpenGL Support

Open Source

Cons:

iOS support costs money

Low level

iOS emulator support

http://libgdx.badlogicgames.com/

Unity 3D

Pros:

Awesome 2D/3D support

Pre-fabbed content for fast game development

Support for 1st or 3rd person

Asset store

Cons:

Costly

High learning curve with tools and APIs

http://unity3d.com/

Cocos 2d

Pros:

Large developer community

Free

Open Source

C++, Lua and Javascript

Cons:

C++

May be complex for some beginners

http://www.cocos2d-x.org/

Corona SDK

Pros:

Easy to display and animate graphics

Tap listeners are easy to implement

Code compiles to native language

Native hardware support

Large developer community

Simulator Support

Cons:

Not free - although affordable

Need to learn Lua

One developer/computer

http://coronalabs.com/