Project Sound Architectural Specification

Anthony Chen, Detian Shi, Leina Sha, Danielle Lee, Natalie Diebold, Nicholas Beaumont

Note: The following architecture uses the entity-component pattern.

Class Responsibility Collaboration Tables

Components

PhysicsPlugin

Description: The PhysicsPlugin class contains physics-related information, and interacts with other PhysicsPlugins. This class serves as the base class for every other PhysicsPlugin.

Justification: All Physics-based objects have a base set of methods and values that dictate their actions during collisions with each other and with pings, so they are defined here to be overridden later.

Classification: Component

Responsibility	Collaboration
Update physics parameters	Box2D, Farseer
Collide	Box2D, Farseer

ControllerPlugin (abstract)

Description: The ControllerPlugin class contains a framework that allows control over a GameEntity. This class serves as the base class for every other ControllerPlugin

Justification: All ControllerPlugins supply an action to the GameEntity, so this class defines available methods to be overwritten.

Classification: Component

Responsibility	Collaboration
Modify Entity State	N/A
Receive ping of some intensity	N/A

ControllerPlugin::PingCP

Description: The PingCP class contains the calculations for an expanding ping.

Justification: The ping entity behaves uniquely to every other object in game.

Classification: Controller Component

Responsibility	Collaboration
Expand	PhysicsPlugin
Collide with objects	GameLevel, GameEntity

ControllerPlugin::PlayerCP

Description: The PlayerCP subclass contains necessary code for controlling the player through keyboard and gamepad interfaces.
Justification: Players must be able to control the hero

Classification: Controller Component

Responsibility	Collaboration
Move	GameEntity
Spawn ping	GameEntity

ControllerPlugin::AIControllerPlugin

Description: The AIPlugin subclass contains all necessary code for processing of the AI. Differentiation in behavior is achieved through subclasses for every distinct AI.

Justification: All NPCs will follow this same AI, so it makes sense to include it as a plugin that can be used by all NPCs.

Classification: Component

Responsibility	Collaboration
Find best paths to target	GameLevel
Find target	GameLevel

GraphicsPlugin

Description: The GraphicsPlugin class contains information regarding the artwork of the given GameEntity. Currently animating frame, Z-index, and other graphical values are all stored here.

Justification: The exact methods for the graphics of two different game objects are very similar, so this can exist as its own plugin, rather than as data unique to each superclass.

Classification: Model Component

Responsibility	Collaboration
Determine/Update current animating frame	N/A
Paint	Canvas

Entities

GameEntity

Description: The Actor class is the physical representation of an intelligent object in the game. It holds an x,y position, tracks actor's remaining sound energy, and collides with other objects.

Justification: All actors use the same components, so they are collected in one class here.

Classification: Model

Responsibility	Collaboration
Delegate physics	PhysicsPlugin
Perform requested action	ControllerPlugin
Delegate Paint call	GraphicsPlugin

GameLevel

Description: The GameLevel class contains all the GameEntity's in a particular level.

Justification: A container class is necessary to keep references to the various entities in the game.

Classification: Model

Responsibility	Collaboration
Update entities in game	GameEntitiy
Contain and modify global physics constants	Box2D, Farseer
Draw GameEntities during paint step	GameEntity

GameEngine

Description: the GameEngine class contains the main update loop and current level.

Justification: A persistent top class is required to allow the game to change levels and views while maintaining state.

Responsibility	Collaboration
Record overall game state	N/A
Run update loop	GameLevel
Delegate Paint call	Canvas
Get user input	PlayerCP

Canvas

Description: The Canvas class is an abstraction of rendering, hiding away animation details

Justification: Prevents the draw method from being bloated and being in game object

Classification: View

Responsibility	Collaboration
Draw all objects	N/A

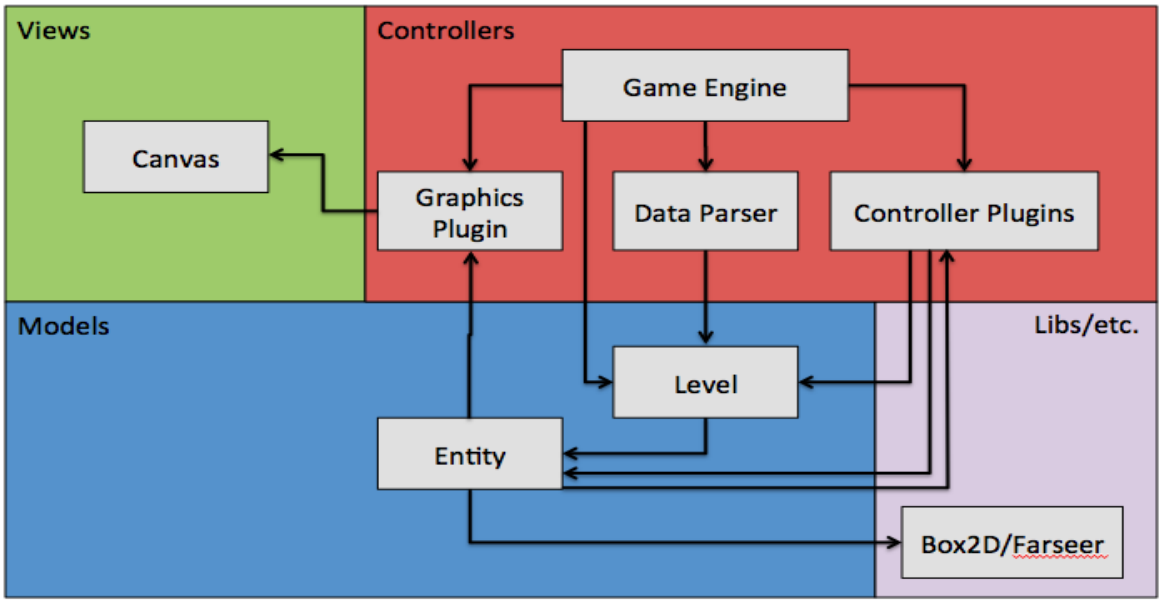
DataParser

Description: the DataParser class reads game level files and creates a GameLevel containing the necessary state.

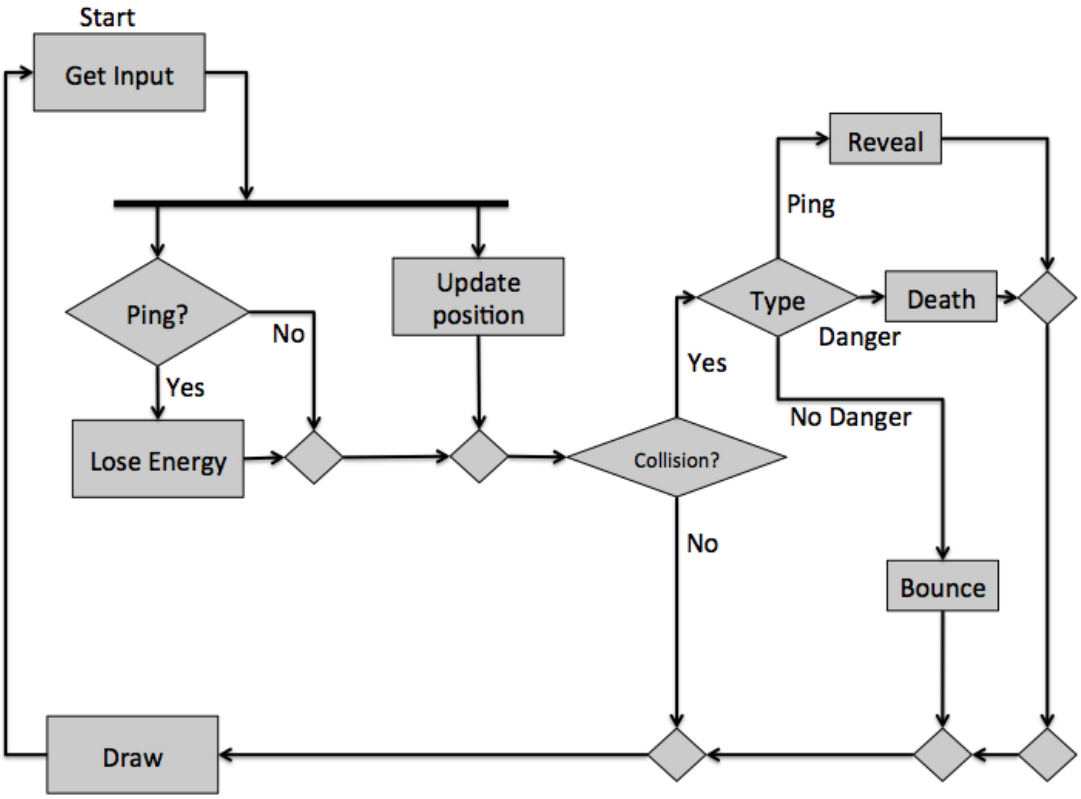
Justification: Need standardized way of saving and loading levels

Responsibility	Collaboration
Load/Save file in game-readable format	GameLevel

Dependency Diagram



Activity Diagram



Data Representation Model

Saved Game File

File Format: XML

Information Stored on File: We will store the list of levels completed, along with the current level.

How Information is Stored: Below is a sample saved game file:

```
<?xml version=”1.0” encoding=”UTF-8” ?>
<LevelsFinished currentLevel=”5”>
      <Level name=”1” state=”complete”/>
      <Level name=”2” state=”complete”/>
      <Level name=”3” state=”complete”/>
      <Level name=”4” state=”complete”/>
      <Level name=”5” state=”incomplete”/>
</LevelsFinished>
```

Level File

File Format: XML

Information Stored on File:

This will store everything necessary to recreate a level. This includes size of the level, difficulty settings, components used (plugins), and assets (such as sprites).

How Information is Stored:

Below is a sample level file with one hunter and one landmark. Note that the ID gives the component being used.

```
<?xml version="1.0" encoding="utf-8" ?>
<Level>
  <MagicNum num="47913277"/>
    <Identifiers name="Tutorial Level" author="RenaS"/>
    <LevelSettings sizex="5" sizey="10" difficulty="(0-9)"/>
<Entities>
      <Entity name="SuperHunter" posx="5" posy="5">        <Controller
ID="Hunter"></Controller>
        <Graphics ID="SuperHunterGP" sprLoc="C:\\funtimes">
        </Graphics>
        <Physics ID="SuperHunterPP">
        </Physics>
      </Entity>
      <Entity name="Landmark1" posx="5" posy="5">
        <Controller ID="NULL">
        </Controller>
        <Graphics ID="LandmarkGP" fileloc="C:\\funtimes">
        </Graphics>
        <Physics ID="LandmarkPP">
        </Physics>
      </Entity>
  </Entities>
</Level>
```