DESIGN PROCESS FOR PORTAL 2D

Mathew Blair (mblair), James Fitzsimmons (fjr), Joana Lopez, (joanal), Pongsakorn Champ Kanjanabutr (pkanj) SWEN30006 - Software Modeling and Design

17th April 2013

This report aims to clarify the design choices made by our team, and explain ambiguous statements that cannot be accurately presented in diagram form.

In this report we will discuss the analysis phase as well as the design decisions made from our findings in the analysis phase. The analysis phase begun with a discussion aiming to scope out the requirements of the product, namely a working physics engine that can maintain momentum and a flexible backend to allow easy addition and customisation of various resources and options.

In order to flesh out these ideas we began with the analysis models, starting with the Domain Model (Figure 1) which allowed us to separate each constituent part into it's own domain and begin to visualize the associations and dependencies present in the model. We then decided to add an additional model that isn't quite standard with the creation of an association map. This map (Figure 2), explains the associations that each object has to each other, allowing us to realize the attributes for each individual class. This clarification created a simple categorization of functions belonging to each object and allowed a seamless transition to design diagrams.

The use-case model (Figure 3) is fairly self explanatory and covers the major use cases involved in this project, which is the navigation of the game menu and associated states. To represent the in-game interactions, we opted for an activity diagram (Figure 4) that shows the flow of control during each update and render state.

The final decision made in the analysis phase was to build on Mat's work from previous subjects and utilize an XML resource loader that has already been implemented following a content-server model. This provided us with a solid foundation to simply modify the existing code to deal with attributes that we require, allowing us to load all the images, sounds and animations as well as XML files for levels at the start of the game, cutting down on loading times later and providing an easy option to get any resource by simply stating:

AssetManager.requestResource(resourceId).

The use of the resource manager in this way allows quick, flexible modifications to be applied prototypes and to the final product. It also allows code flexibility as fewer variables are hard coded, allowing for such things as dynamic input modifications and easy addition of new levels and assets.

Another decision that we have made in the analysis process, which isn't really easy to demonstrate through diagrams, is the choice to use JBox2D (the Java port of Box2D) as our physics engine. Given the complex nature of our world, we need to have a solid physics engine that will allow all objects to interact according to the players expectations (i.e. as in real life). The realism afforded to us by a complete physics simulation is crucial to the successful teaching of momentum, as we must be able to demonstrate concepts such as the conservation of momentum, applying impulses and force interactions between various bodies.

Writing this engine ourselves would have been a massive undertaking, requiring a project much larger in scope than the one we are attempting now, implementing a differential equation based physics simulator that updates pre- and post-solve positions, calculating all body positions and doing so quickly enough so as to not cause lag in gameplay. Given that the project is testing our design skills - and more importantly, how intelligent our design is - the intelligent decision was made to use an off the shelf open-source implementation

that is considered class leading, allowing us to focus on delivering a more polished and refined game experience.

The design phase began with two interaction diagrams, one for loading levels (Figure 5) and one for loading achievements (Figure 6). We chose to do these two diagrams as they are most crucial to understanding and implementing our design. The achievement loading will create the template that high-score loading will follow, so getting this design right is imperative to our success in implementation. This diagram should be straightforward to follow; the only thing to be noted is that we have decided to omit the update and render steps of the interaction flow as they are not crucial to the steps that we are trying to design.

The second interaction diagram was done to clarify the method by which we load levels from XML files before entering the game state. Again, this should be quite self explanatory and was translated quite readily to the design class diagrams.

For the class diagrams we have chosen to split the system up into three separate diagrams in order to facilitate a more cursory reading. The first of these three (Figure 7) describes the general overview of the Game Engine and the associated classes responsible for the management of gameplay. We have chosen, for legibility, to leave out the inherited parts of the API that are used from Slick2D and JBox2D. This includes the extension of super objects for the game container, as well as the OpenGL graphics that Slick is using under the hood. As these are API functions, we assume that if any query is made about them, the querying party will be able to look up the java-doc to ascertain their function.

The second class diagram (Figure 8) describes the in-game state, and the interactions that occur between each object. This shows the attributes that levels have, how different objects will interact with each other and the methods available on each object. The final class diagram (Figure 9) demonstrates the function of the resource loader, managed by AssetManager, and how each resource loading class is interacted with, called and what the requests for assets return.

These design models, as far as we see, should be complete and representative of the final product (bar any minor changes made to facilitate implementation or any changes made after finding more efficient ways to do things). We should note that the design process does not stop at implementation; instead we will be constantly taking a step back and looking at what we have so far, running tests to determine its efficiency and if we can see a better method of implementing something: it would be pointless to toss a more efficient solution aside in favour of sticking to the class diagram.

Given the unfamiliarity of the JBox2D engine, we anticipate that it is possible that we may find more efficient methods to deal with the various physics actions, particularly the creation and linking of portals. Unfortunately, a large majority of the documentation for JBox2D is incomplete, and there are many functions that accomplish the same thing but give no information as to their speed or efficiency. As a result, we have made sure to note that we attempt the various ways of solving problems and determine the best based on their performance in a real game situation.

That being said, we do not expect to make any changes to the general layout or design of the Slick StateBasedG ame implementation. All of the relations between objects and the level are final, as are all the interactions and the design of loading levels and loading assets. The only parts that should be subject to change will be the specific functions for moving objects or interacting with the physics world.

In order to facilitate ease of communication we have attached the diagrams to this document labeled with the appropriate figure headings.

Figure 1

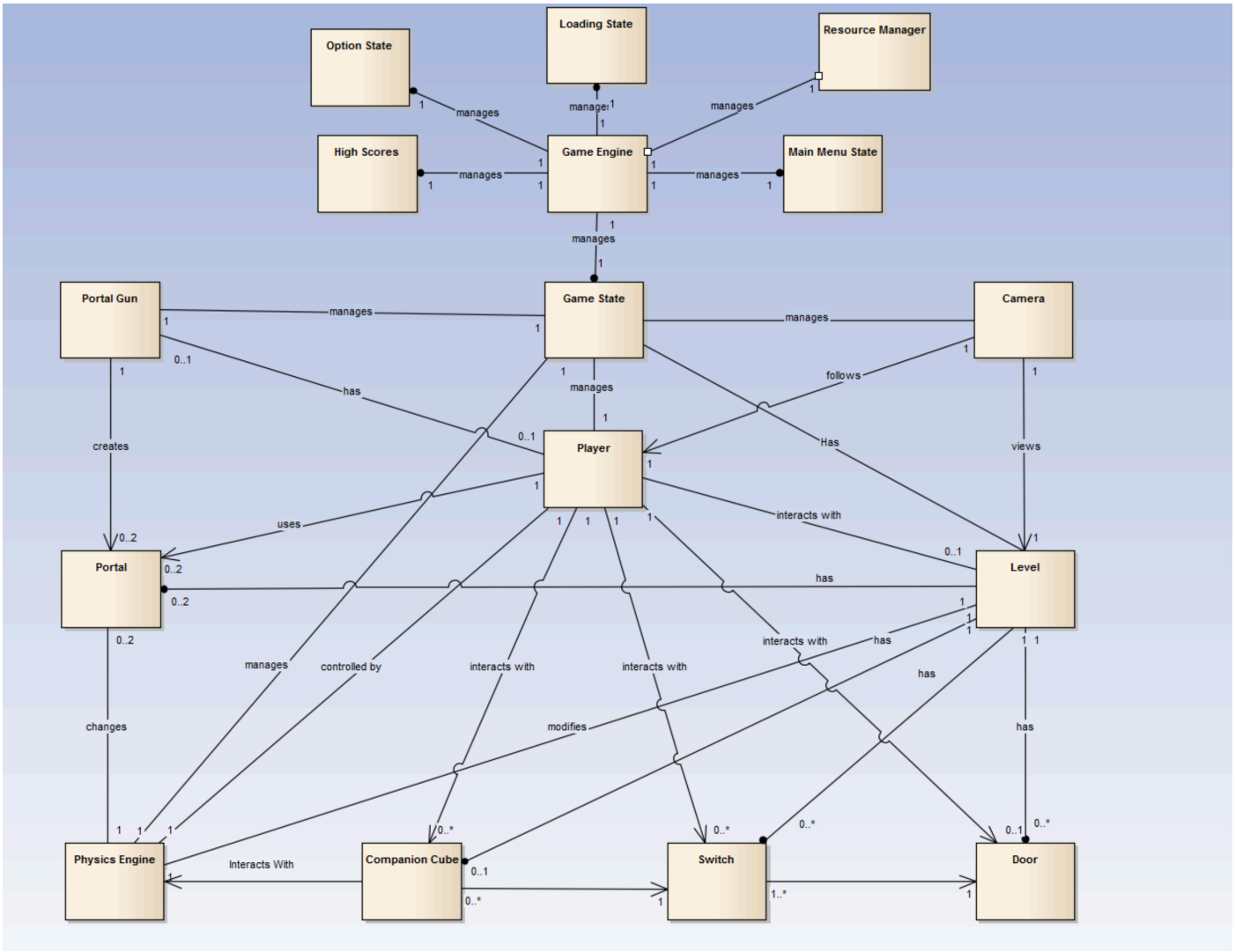

Loading State
Option State
Resource Manager
High Scores
Game Engine
Main Menu State
Portal Gun
Game State
Camera
Player
Portal
Level
Physics Engine
Companion Cube
Switch
Door
manages
has
creates
follows
views
Has
uses
interacts with
changes
controlled by
modifies
Interacts With

Figure 2

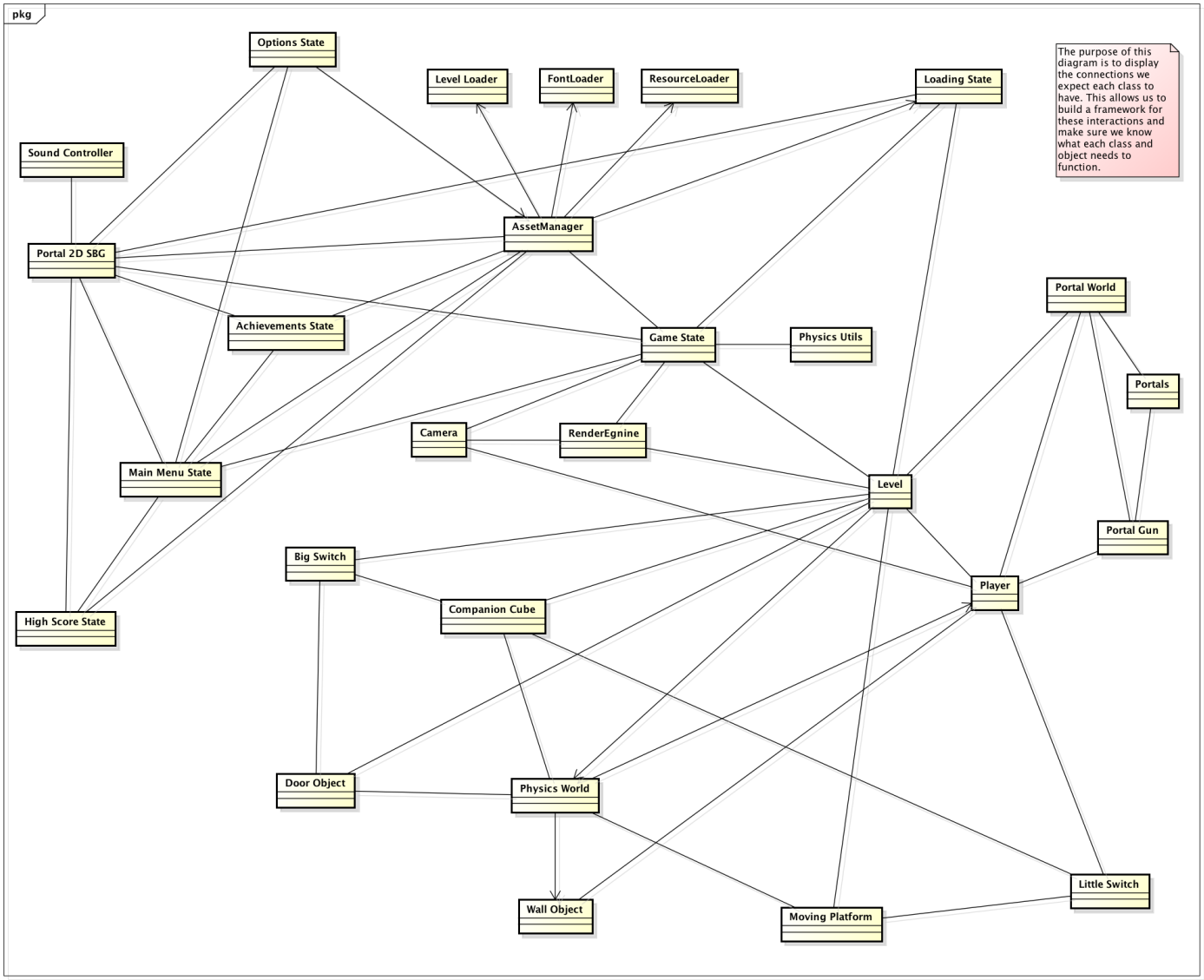
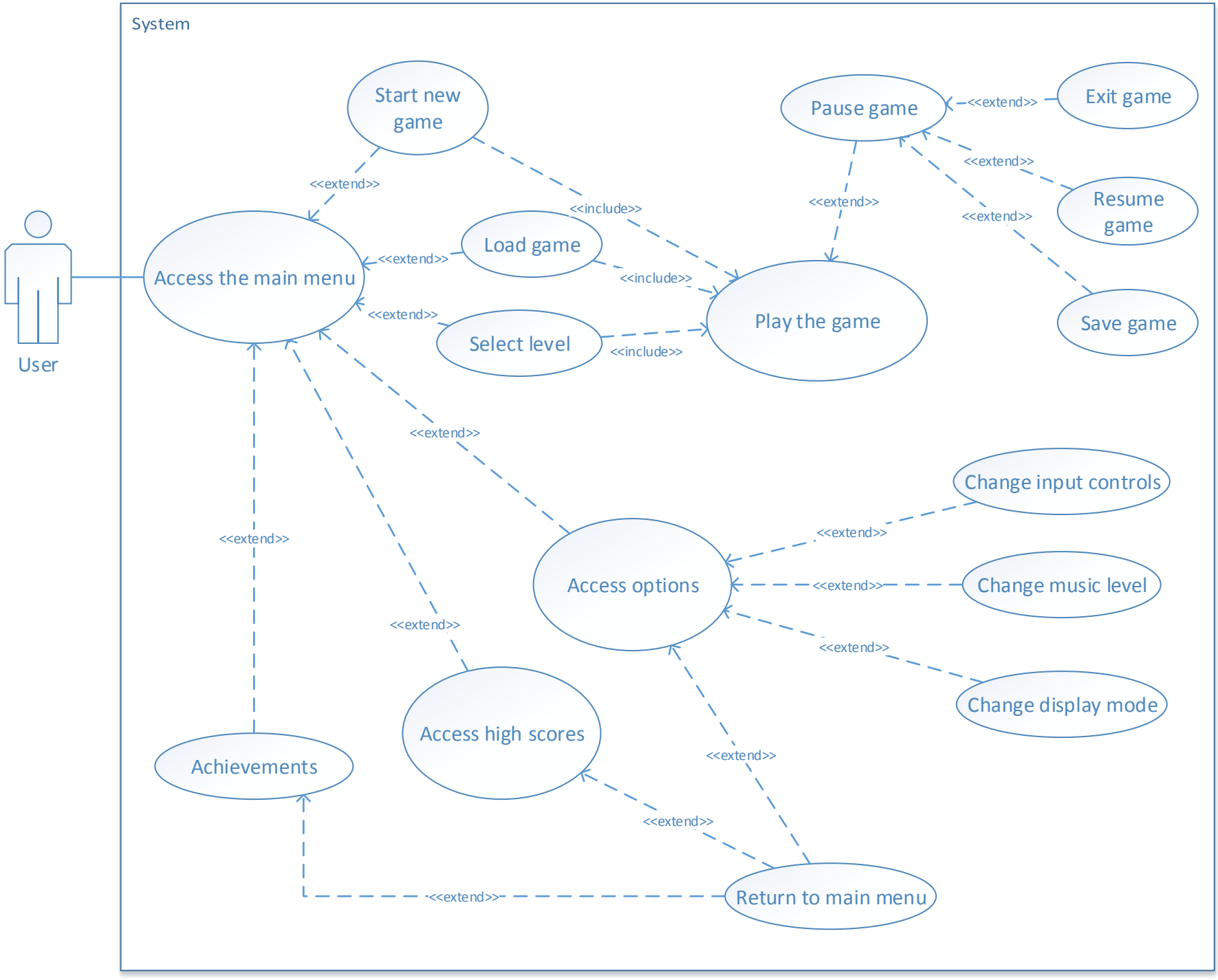

pkg
Options State
Level Loader
FontLoader
ResourceLoader
Loading State
The purpose of this diagram is to display the connections we expect each class to have. This allows us to build a framework for these interactions and make sure we know what each class and object needs to function.
Sound Controller
AssetManager
Portal 2D SBG
Portal World
Achievements State
Game State
Physics Utils
Portals
Camera
RenderEgnine
Main Menu State
Level
Portal Gun
Big Switch
Player
Companion Cube
High Score State
Door Object
Physics World
Little Switch
Wall Object
Moving Platform

Figure 3


System
User
Start new game
Access the main menu
Load game
Select level
Play the game
Pause game
Exit game
Resume game
Save game
Access options
Change input controls
Change music level
Change display mode
Access high scores
Achievements
Return to main menu
<<extend>>
<<include>>

Use Case Descriptions

USE CASE NAME	Access the main menu
BRIEF DESCRIPTION	This Use Case describes the process of a user accessing the main menu of the game
ACTORS	User
STAKEHOLDERS	The user who wants to access the game
PRE-CONDITIONS	None
POST-CONDITIONS	The user will be presented with the main menu view and will be able to choose between the following options: - Start Game - Load Game - Select Level - Options - High Scores
TRIGGERING EVENT	A user wants to access the main menu
BASIC FLOW	1. The user access the main menu view 2. The Use Case ends
ALTERNATE/EXCEPTIONAL FLOWS	None

USE CASE NAME	Start new game
BRIEF DESCRIPTION	This Use Case describes the process of a user starting a new game
ACTORS	User
STAKEHOLDERS	The user who wants to start playing a new game
PRE-CONDITIONS	The user must have accessed the main menu previously
POST-CONDITIONS	The user will be presented with the in game view and will be able to start playing a new

	game from the first level
TRIGGERING EVENT	A user wants to start playing a new game
BASIC FLOW	1. The user clicks the Start New Game option from the main menu 2. The user appears in the game view 3. The Use Case ends
ALTERNATE/EXCEPTIONAL FLOWS	None

USE CASE NAME	Load game
BRIEF DESCRIPTION	This Use Case describes the process of a user loading the saved game
ACTORS	User
STAKEHOLDERS	The user who wants to load the saved game
PRE-CONDITIONS	The user must have accessed the main menu previously and there must be a saved game to load
POST-CONDITIONS	The user will be presented with the in game view and will be able to start playing the game from the level where it was saved previously
TRIGGERING EVENT	A user wants to load the saved game
BASIC FLOW	1. The user clicks the Load Game option from the main menu 2. The user appears in the game view at the saved level 3. The Use Case ends
ALTERNATE/EXCEPTIONAL FLOWS	None

USE CASE NAME	Select level
BRIEF DESCRIPTION	This Use Case describes the process of a user selecting the game level
ACTORS	User

STAKEHOLDERS	The user who wants to select a game level
PRE-CONDITIONS	The user must have accessed the main menu previously, he/she must have completed at least one level to have the "Select Level" activated
POST-CONDITIONS	The user will access the select level view and he/she will be allowed to choose only the completed levels
TRIGGERING EVENT	A user wants to select the game level
BASIC FLOW	1. The user clicks the Select Level option from the main menu 2. The user appears in the select level view 3. The user selects the desired level 4. The user appears in the game view at the selected level 5. The Use Case ends
ALTERNATE/EXCEPTIONAL FLOWS	None

USE CASE NAME	Play the game
BRIEF DESCRIPTION	This Use Case describes the process of a user playing the game
ACTORS	User
STAKEHOLDERS	The user who wants to play the game
PRE-CONDITIONS	The user must have accessed the main menu previously and selected one of the following options: - Start new game - Load game - Select level
POST-CONDITIONS	The user will play the game as long as he/she wants or until he/she completes the whole game
TRIGGERING EVENT	A user wants to play the game
BASIC FLOW	Steps 1 to 5 don't need to be sequential

	1. The user explores the map moves around in order to get an idea of the map 2. The user can jump and move horizontally to clear un-passable hurdles 3. The user creates a pair of portals using a portal device 4. The user needs to deal with possible enemies 5. The user needs to reach the box and bring it to the switch 6. The user goes to the next level 7. Go back to step 1 while more levels exist 8. The Use Case ends
ALTERNATE/EXCEPTIONAL FLOWS	The user pauses the game while playing. Go to use case "Pause game"

USE CASE NAME	Pause game
BRIEF DESCRIPTION	This Use Case describes the process of a user pausing the game while playing
ACTORS	User
STAKEHOLDERS	The user who wants to pause the game
PRE-CONDITIONS	The user must be playing
POST-CONDITIONS	The game will be paused and the user will be able to select from the following options: - Save game - Resume gam - Exit game
TRIGGERING EVENT	A user wants to pause the game
BASIC FLOW	1. The user clicks the Pause option while playing 2. The game is paused and the options are shown to the user 3. The Use Case ends
ALTERNATE/EXCEPTIONAL FLOWS	None

USE CASE NAME	Save game
BRIEF DESCRIPTION	This Use Case describes the process of a user saving the current game
ACTORS	User
STAKEHOLDERS	The user who wants to save the game
PRE-CONDITIONS	The user must have paused the game
POST-CONDITIONS	The game will be saved and the user will be able to select from the following options: - Exit game - Resume game - Save game
TRIGGERING EVENT	A user wants to save the game
BASIC FLOW	1. The user clicks the Save game option 2. The game is saved 3. The Use Case ends
ALTERNATE/EXCEPTIONAL FLOWS	None

USE CASE NAME	Resume game
BRIEF DESCRIPTION	This Use Case describes the process of a user resuming the game
ACTORS	User
STAKEHOLDERS	The user who wants to resume the game
PRE-CONDITIONS	The user must have paused the game
POST-CONDITIONS	The game will be resumed and the user will be able to keep on playing
TRIGGERING EVENT	A user wants to resume the game
BASIC FLOW	1. The user clicks the Resume game option 2. The game is resumed and the user is able to keep on playing

	3. The Use Case ends
ALTERNATE/EXCEPTIONAL FLOWS	None

USE CASE NAME	Exit game
BRIEF DESCRIPTION	This Use Case describes the process of a user exiting the game
ACTORS	User
STAKEHOLDERS	The user who wants to exit the game
PRE-CONDITIONS	The user must have paused the game
POST-CONDITIONS	The game will be exited and the user will go back to the main menu
TRIGGERING EVENT	A user wants to exit the game
BASIC FLOW	1. The user clicks the Exit game option 2. The game is exited and the user goes back to the main menu 3. The Use Case ends
ALTERNATE/EXCEPTIONAL FLOWS	None

USE CASE NAME	Access options
BRIEF DESCRIPTION	This Use Case describes the process of a user accessing the options of the game
ACTORS	User
STAKEHOLDERS	The user who wants to change the options
PRE-CONDITIONS	None
POST-CONDITIONS	The user will be presented with the different options that are possible to change: - Input controls - Music level - Display modes

TRIGGERING EVENT	A user wants to access the options of the game
BASIC FLOW	1. The user access the options 2. The Use Case ends
ALTERNATE/EXCEPTIONAL FLOWS	None

USE CASE NAME	Change input controls
BRIEF DESCRIPTION	This Use Case describes the process of a user changing the input controls of the game
ACTORS	User
STAKEHOLDERS	The user who wants to change the input controls
PRE-CONDITIONS	The user must have accessed the options view previously
POST-CONDITIONS	The user will be able to change the input controls
TRIGGERING EVENT	A user wants to change the input controls
BASIC FLOW	1. The user chooses to change input controls 2. The user changes the desire controls 3. The Use Case ends
ALTERNATE/EXCEPTIONAL FLOWS	None

USE CASE NAME	Change music level
BRIEF DESCRIPTION	This Use Case describes the process of a user changing the music level of the game
ACTORS	User

STAKEHOLDERS	The user who wants to change the music level
PRE-CONDITIONS	The user must have accessed the options view previously
POST-CONDITIONS	The user will be able to change the music level
TRIGGERING EVENT	A user wants to change the music level
BASIC FLOW	1. The user chooses to change the music level 2. The user changes the volume 3. The Use Case ends
ALTERNATE/EXCEPTIONAL FLOWS	None

USE CASE NAME	Change display mode
BRIEF DESCRIPTION	This Use Case describes the process of a user changing the display mode of the game
ACTORS	User
STAKEHOLDERS	The user who wants to change the display mode
PRE-CONDITIONS	The user must have accessed the options view previously
POST-CONDITIONS	The user will be able to change the display mode
TRIGGERING EVENT	A user wants to change the display mode
BASIC FLOW	1. The user chooses to change the display mode 2. The user changes the display mode 3. The Use Case ends
ALTERNATE/EXCEPTIONAL FLOWS	None

USE CASE NAME	Access high scores
BRIEF DESCRIPTION	This Use Case describes the process of a user accessing the high scores of the game
ACTORS	User
STAKEHOLDERS	The user who wants to see the high scores
PRE-CONDITIONS	None
POST-CONDITIONS	The user will be presented with the view of the high scores ordered from the highest to the lowest score
TRIGGERING EVENT	A user wants to access the high scores of the game
BASIC FLOW	1. The user access the high scores 2. The Use Case ends
ALTERNATE/EXCEPTIONAL FLOWS	None

USE CASE NAME	Access achievements
BRIEF DESCRIPTION	This Use Case describes the process of a user accessing the achievements
ACTORS	User
STAKEHOLDERS	The user who wants to see the achievements
PRE-CONDITIONS	None
POST-CONDITIONS	The user will be presented with the view of the achievements
TRIGGERING EVENT	A user wants to access the achievements
BASIC FLOW	1. The user access the achievements 2. The Use Case ends
ALTERNATE/EXCEPTIONAL FLOWS	None

USE CASE NAME	Return to the main menu
BRIEF DESCRIPTION	This Use Case describes the process of a user returning to the main menu
ACTORS	User
STAKEHOLDERS	The user who wants to return to the main menu
PRE-CONDITIONS	The user must be in either Options, High scores or Achievements
POST-CONDITIONS	The user will return to the main menu
TRIGGERING EVENT	A user wants to return to the main menu
BASIC FLOW	1. The user clicks the Return to the main menu option 2. The user goes back to the main menu 3. The Use Case ends
ALTERNATE/EXCEPTIONAL FLOWS	None

Figure 4

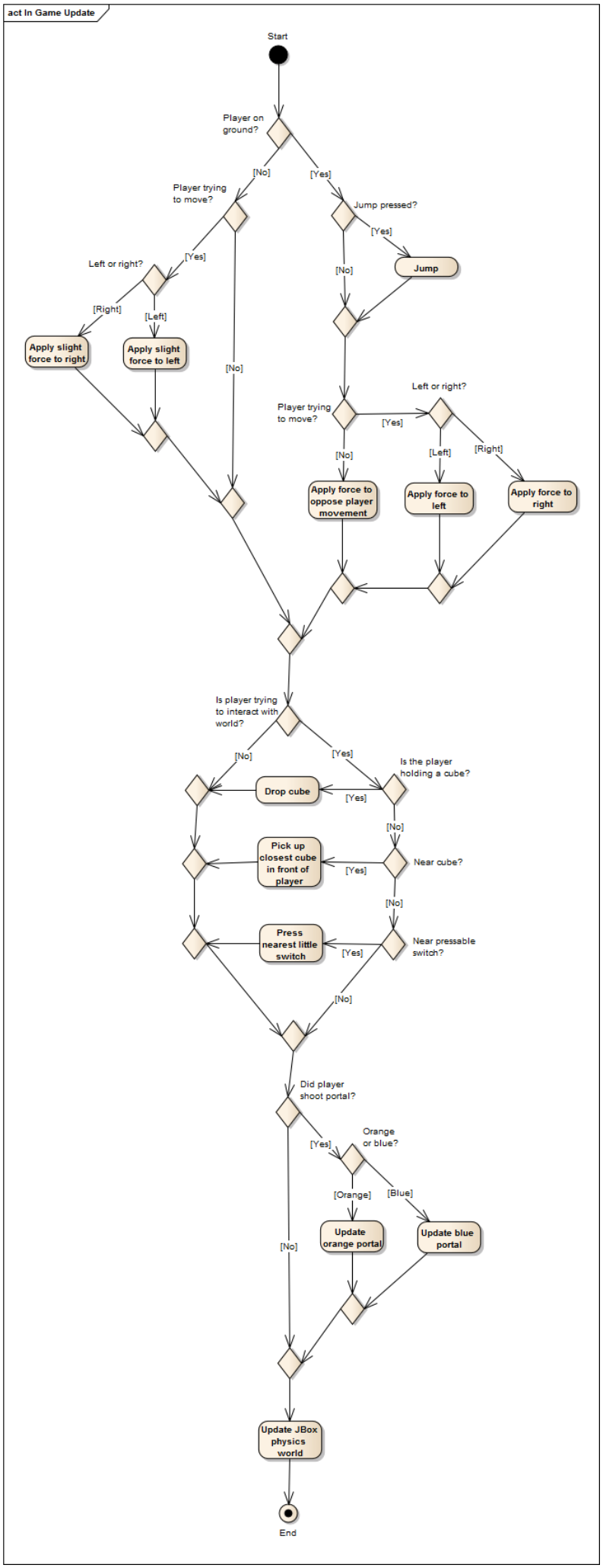

act In Game Update
Start
Player on ground?
[No]
[Yes]
Player trying to move?
Jump pressed?
[Yes]
[Yes]
[No]
Jump
Left or right?
[Right]
[Left]
Apply slight force to right
Apply slight force to left
[No]
Player trying to move?
Left or right?
[Yes]
[No]
[Left]
[Right]
Apply force to oppose player movement
Apply force to left
Apply force to right
Is player trying to interact with world?
[No]
[Yes]
Is the player holding a cube?
Drop cube
[Yes]
[No]
Pick up closest cube in front of player
[Yes]
Near cube?
[No]
Press nearest little switch
[Yes]
Near pressable switch?
[No]
Did player shoot portal?
[Yes]
Orange or blue?
[Orange]
[Blue]
Update orange portal
Update blue portal
[No]
Update JBox physics world
End

Figure 5

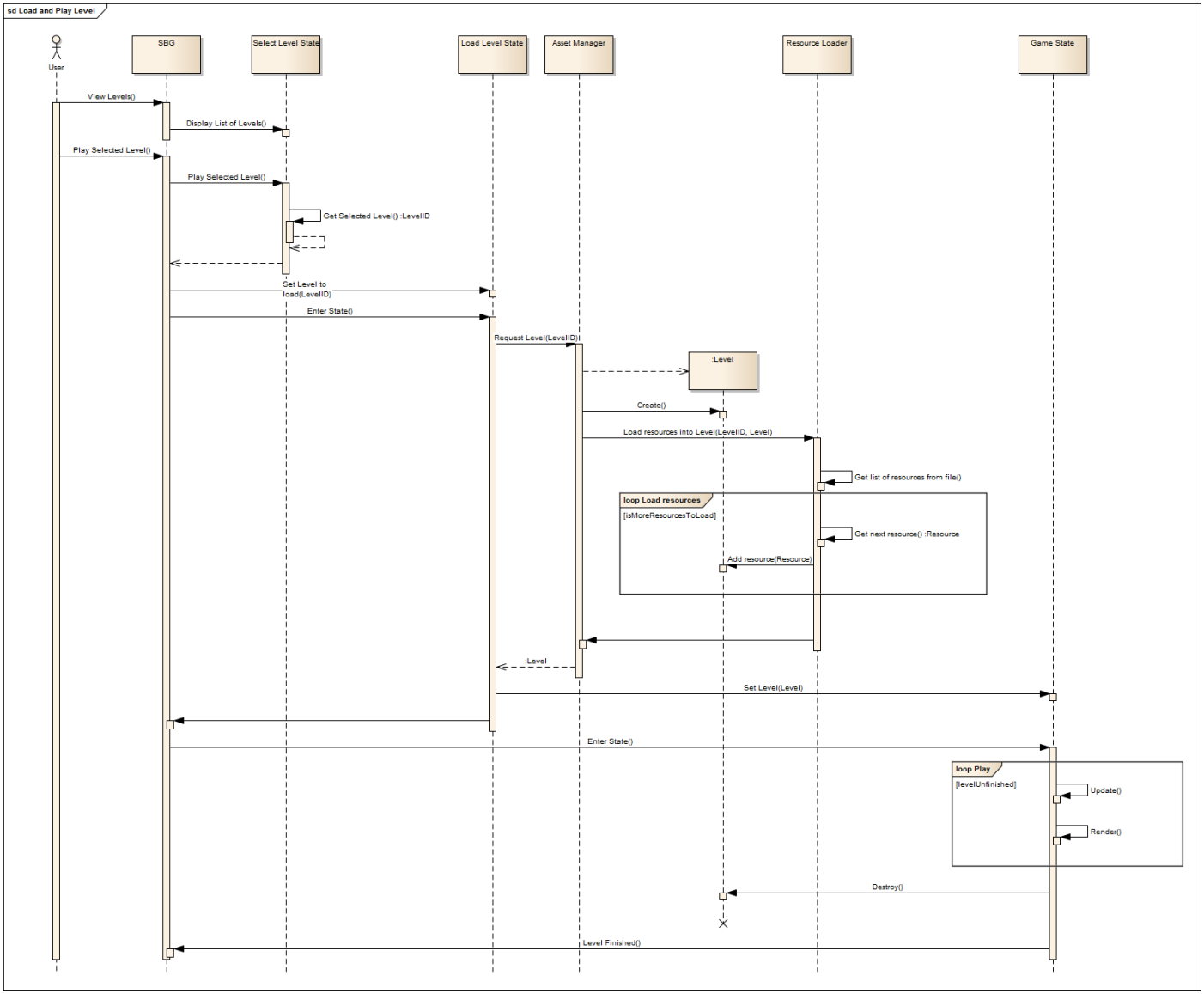

sd Load and Play Level
User
SBG
Select Level State
Load Level State
Asset Manager
Resource Loader
Game State
View Levels()
Display List of Levels()
Play Selected Level()
Play Selected Level()
Get Selected Level() :LevelID
Set Level to load(LevelID)
Enter State()
Request Level(LevelID)
:Level
Create()
Load resources into Level(LevelID, Level)
Get list of resources from file()
loop Load resources
[isMoreResourcesToLoad]
Get next resource() :Resource
Add resource(Resource)
:Level
Set Level(Level)
Enter State()
loop Play
[levelUnfinished]
Update()
Render()
Destroy()
Level Finished()

Figure 6

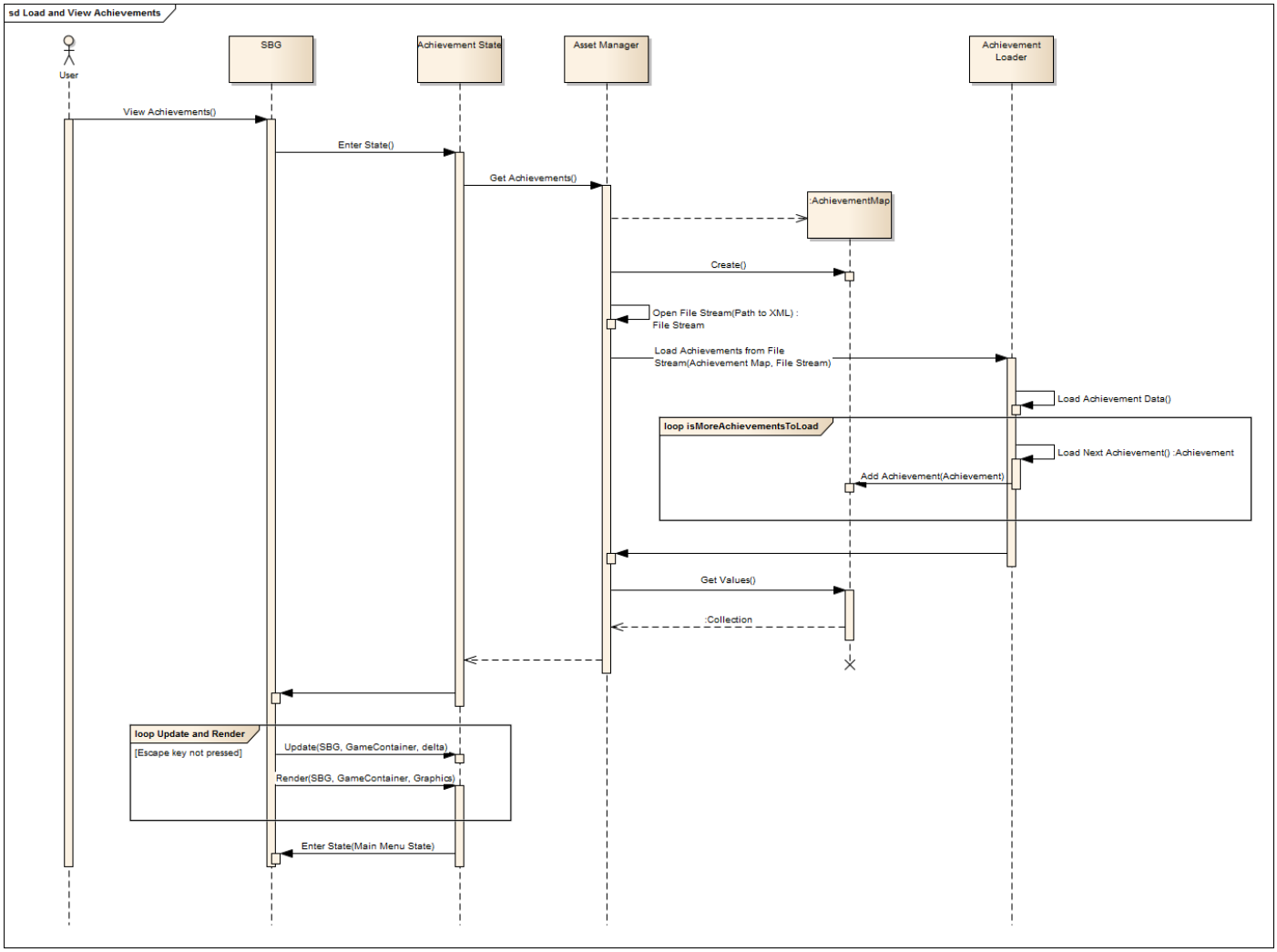

sd Load and View Achievements
User
SBG
Achievement State
Asset Manager
:AchievementMap
Achievement Loader
View Achievements()
Enter State()
Get Achievements()
Create()
Open File Stream(Path to XML) : File Stream
Load Achievements from File Stream(Achievement Map, File Stream)
Load Achievement Data()
loop isMoreAchievementsToLoad
Load Next Achievement() :Achievement
Add Achievement(Achievement)
Get Values()
:Collection
loop Update and Render
[Escape key not pressed]
Update(SBG, GameContainer, delta)
Render(SBG, GameContainer, Graphics)
Enter State(Main Menu State)

Figure 7

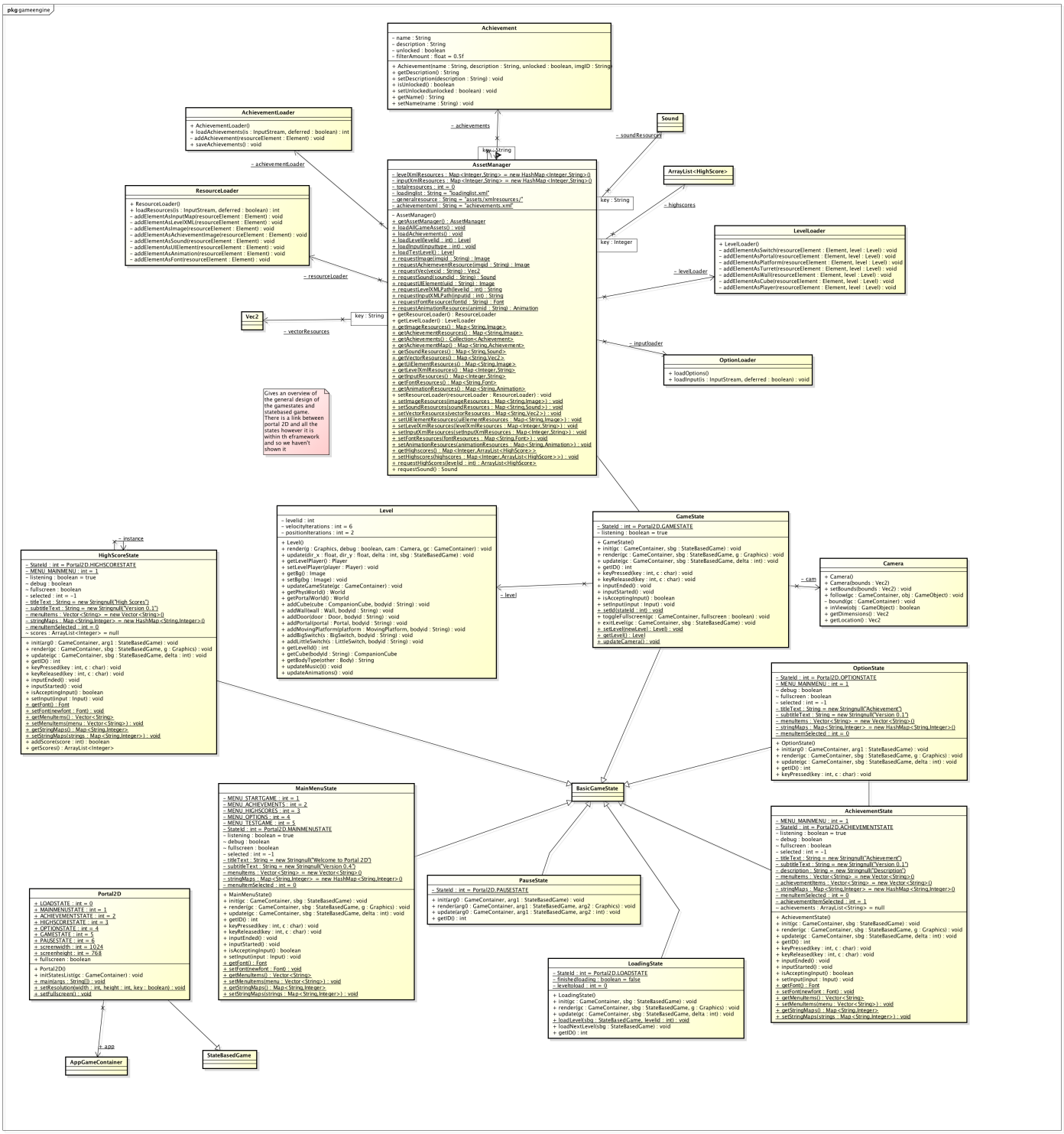

pkg gameengine
Achievement
- name : String
- description : String
- unlocked : boolean
- filterAmount : float = 0.5f
+ Achievement(name : String, description : String, unlocked : boolean, imgID : String)
+ getDescription() : String
+ setDescription(description : String) : void
+ isUnlocked() : boolean
+ setUnlocked(unlocked : boolean) : void
+ getName() : String
+ setName(name : String) : void
- achievements
AchievementLoader
+ AchievementLoader()
+ loadAchievements(is : InputStream, deferred : boolean) : int
- addAchievement(resourceElement : Element) : void
+ saveAchievements() : void
- achievementLoader
Sound
- soundResources
key : String
ArrayList<HighScore>
- highscores
ResourceLoader
+ ResourceLoader()
+ loadResources(is : InputStream, deferred : boolean) : int
- addElementAsInputMap(resourceElement : Element) : void
- addElementAsLevelXML(resourceElement : Element) : void
- addElementAsImage(resourceElement : Element) : void
- addElementAsAchievementImage(resourceElement : Element) : void
- addElementAsSound(resourceElement : Element) : void
- addElementAsUIElement(resourceElement : Element) : void
- addElementAsAnimation(resourceElement : Element) : void
- addElementAsFont(resourceElement : Element) : void
- resourceLoader
AssetManager
- levelXmlResources : Map<Integer,String> = new HashMap<Integer,String>()
- inputXmlResources : Map<Integer,String> = new HashMap<Integer,String>()
- totalresources : int = 0
- loadinglist : String = "loadinglist.xml"
- generalresource : String = "assets/xmlresources/"
- achievementsxml : String = "achievements.xml"
- AssetManager()
+ getAssetManager() : AssetManager
+ loadAllGameAssets() : void
+ loadAchievements() : void
+ loadLevel(levelid : int) : Level
+ loadInput(inputtype : int) : void
+ loadTextLevel() : Level
+ requestImage(imgid : String) : Image
+ requestAchievementResource(imgid : String) : Image
+ requestVec(vecid : String) : Vec2
+ requestSound(soundid : String) : Sound
+ requestUIElement(uiid : String) : Image
+ requestLevelXMLPath(levelid : int) : String
+ requestInputXMLPath(inputid : int) : String
+ requestFontResource(fontid : String) : Font
+ requestAnimationResources(animid : String) : Animation
+ getResourceLoader() : ResourceLoader
+ getLevelLoader() : LevelLoader
+ getImageResources() : Map<String,Image>
+ getAchievementResources() : Map<String,Image>
+ getAchievements() : Collection<Achievement>
+ getAchievementMap() : Map<String,Achievement>
+ getSoundResources() : Map<String,Sound>
+ getVectorResources() : Map<String,Vec2>
+ getUiElementResources() : Map<String,Image>
+ getLevelXmlResources() : Map<Integer,String>
+ getInputResources() : Map<Integer,String>
+ getFontResources() : Map<String,Font>
+ getAnimationResources() : Map<String,Animation>
+ setResourceLoader(resourceLoader : ResourceLoader) : void
+ setImageResources(imageResources : Map<String,Image>) : void
+ setSoundResources(soundResources : Map<String,Sound>) : void
+ setVectorResources(vectorResources : Map<String,Vec2>) : void
+ setUIElementResources(uiElementResources : Map<String,Image>) : void
+ setLevelXmlResources(levelXmlResources : Map<Integer,String>) : void
+ setInputXmlResources(setInputXmlResources : Map<Integer,String>) : void
+ setFontResources(fontResources : Map<String,Font>) : void
+ setAnimationResources(animationResources : Map<String,Animation>) : void
+ getHighscores() : Map<Integer,ArrayList<HighScore>>
+ setHighscore(highscores : Map<Integer,ArrayList<HighScore>>) : void
+ requestHighScores(levelid : int) : ArrayList<HighScore>
+ requestSound() : Sound
key : Integer
- levelLoader
LevelLoader
+ LevelLoader()
- addElementAsPortal(resourceElement : Element, level : Level) : void
- addElementAsPlatform(resourceElement : Element, level : Level) : void
- addElementAsTurret(resourceElement : Element, level : Level) : void
- addElementAsWall(resourceElement : Element, level : Level) : void
- addElementAsCube(resourceElement : Element, level : Level) : void
- addElementAsPlayer(resourceElement : Element, level : Level) : void
Vec2
key : String
- vectorResources
- inputloader
OptionLoader
+ loadOptions()
+ loadInput(is : InputStream, deferred : boolean) : void
Gives an overview of the general design of the gamestates and statebased game.
There is a link between portal 2D and all the states however it is within th eframework and so we haven't shown it
Level
- levelid : int
- velocityIterations : int = 6
- positionIterations : int = 2
+ Level()
+ render(g : Graphics, debug : boolean, cam : Camera, gc : GameContainer) : void
+ update(dir_x : float, dir_y : float, delta : int, sbg : StateBasedGame) : void
+ getLevelPlayer() : Player
+ setLevelPlayer(player : Player) : void
+ getBg() : Image
+ setBg(bg : Image) : void
+ updateGameState(gc : GameContainer) : void
+ getPhysWorld() : World
+ getPortalWorld() : World
+ addCube(cube : CompanionCube, bodyid : String) : void
+ addWall(wall : Wall, bodyid : String) : void
+ addDoor(door : Door, bodyid : String) : void
+ addPortal(portal : Portal, bodyid : String) : void
+ addMovingPlatform(platform : MovingPlatform, bodyid : String) : void
+ addBigSwitch(s : BigSwitch, bodyid : String) : void
+ addLittleSwitch(s : LittleSwitch, bodyid : String) : void
+ getLevelid() : int
+ getCube(bodyId : String) : CompanionCube
+ getBodyType(other : Body) : String
+ updateMusic() : void
+ updateAnimations() : void
- level
GameState
- StateId : int = Portal2D.GAMESTATE
- listening : boolean = true
+ GameState()
+ init(gc : GameContainer, sbg : StateBasedGame) : void
+ render(gc : GameContainer, sbg : StateBasedGame, g : Graphics) : void
+ update(gc : GameContainer, sbg : StateBasedGame, delta : int) : void
+ getID() : int
+ keyPressed(key : int, c : char) : void
+ keyReleased(key : int, c : char) : void
+ inputEnded() : void
+ inputStarted() : void
+ isAcceptingInput() : boolean
+ setInput(input : Input) : void
+ setId(stateId : int) : void
+ toggleFullscreen(gc : GameContainer, fullscreen : boolean) : void
+ exitLevel(gc : GameContainer, sbg : StateBasedGame) : void
+ setLevel(next_level : Level) : void
+ getLevel() : Level
+ updateCamera() : void
- cam
Camera
+ Camera()
+ Camera(bounds : Vec2)
+ setBounds(bounds : Vec2) : void
+ follow(gc : GameContainer, obj : GameObject) : void
- bound(gc : GameContainer) : void
+ inView(obj : GameObject) : boolean
+ getDimensions() : Vec2
+ getLocation() : Vec2
- instance
HighScoreState
- StateId : int = Portal2D.HIGHSCORESTATE
- MENU_MAINMENU : int = 1
- listening : boolean = true
- debug : boolean
- fullscreen : boolean
- selected : int = -1
- titleText : String = new Stringnull("High Scores")
- subtitleText : String = new Stringnull("Version 0.1")
- menuItems : Vector<String> = new Vector<String>()
- stringMaps : Map<String,Integer> = new HashMap<String,Integer>()
- menuItemSelected : int = 0
~ scores : ArrayList<Integer> = null
+ initIargO : GameContainer, arg1 : StateBasedGame) : void
+ render(gc : GameContainer, sbg : StateBasedGame, g : Graphics) : void
+ update(gc : GameContainer, sbg : StateBasedGame, delta : int) : void
+ getID() : int
+ keyPressed(key : int, c : char) : void
+ keyReleased(key : int, c : char) : void
+ inputEnded() : void
+ inputStarted() : void
+ isAcceptingInput() : boolean
+ setInput(input : Input) : void
+ getFont() : Font
+ setFont(newFont : Font) : void
+ getMenuItems() : Vector<String>
+ setMenuItems(menu : Vector<String>) : void
+ getStringMaps() : Map<String,Integer>
+ setStringMaps(strings : Map<String,Integer>) : void
+ addScores(score : int) : boolean
+ getScores() : ArrayList<Integer>
OptionState
- StateId : int = Porta2D.OPTIONSTATE
- MENU_MAINMENU : int = 1
~ debug : boolean
~ fullscreen : boolean
~ selected : int = -1
- titleText : String = new Stringnull("Achievement")
- subtitleText : String = new Stringnull("Version 0.1")
- menuItems : Vector<String> = new Vector<String>()
- stringMaps : Map<String,Integer> = new HashMap<String,Integer>()
- menuItemSelected : int = 0
+ OptionState()
+ initIargO : GameContainer, arg1 : StateBasedGame) : void
+ render(gc : GameContainer, sbg : StateBasedGame, g : Graphics) : void
+ update(gc : GameContainer, sbg : StateBasedGame, delta : int) : void
+ getID() : int
+ keyPressed(key : int, c : char) : void
BasicGameState
MainMenuState
- MENU_STARTGAME : int = 1
- MENU_ACHIEVEMENTS : int = 2
- MENU_HIGHSCORES : int = 3
- MENU_OPTIONS : int = 4
- MENU_TESTGAME : int = 5
- StateId : int = Portal2D.MAINMENUSTATE
~ listening : boolean = true
~ debug : boolean
~ fullscreen : boolean
~ selected : int = -1
- titleText : String = new Stringnull("Welcome to Portal 2D")
- subtitleText : String = new Stringnull("Version 0.4")
- menuItems : Vector<String> = new Vector<String>()
- stringMaps : Map<String,Integer> = new HashMap<String,Integer>()
- menuItemSelected : int = 0
+ MainMenuState()
+ init(gc : GameContainer, sbg : StateBasedGame) : void
+ render(gc : GameContainer, sbg : StateBasedGame, g : Graphics) : void
+ update(gc : GameContainer, sbg : StateBasedGame, delta : int) : void
+ getID() : int
+ keyPressed(key : int, c : char) : void
+ keyReleased(key : int, c : char) : void
+ inputEnded() : void
+ inputStarted() : void
+ isAcceptingInput() : boolean
+ setInput(input : Input) : void
+ getFont() : Font
+ setFont(newFont : Font) : void
+ getMenuItems() : Vector<String>
+ setMenuItems(menu : Vector<String>) : void
+ getStringMaps() : Map<String,Integer>
+ setStringMaps(strings : Map<String,Integer>) : void
PauseState
- StateId : int = Portal2D.PAUSESTATE
+ initIargO : GameContainer, arg1 : StateBasedGame) : void
+ render(arg0 : GameContainer, arg1 : StateBasedGame, arg2 : Graphics) : void
+ update(arg0 : GameContainer, arg1 : StateBasedGame, arg2 : int) : void
+ getID() : int
AchievementState
- MENU_MAINMENU : int = 1
- StateId : int = Portal2D.ACHIEVEMENTSTATE
~ listening : boolean = true
~ debug : boolean
~ fullscreen : boolean
~ selected : int = -1
- titleText : String = new Stringnull("Achievement")
- subtitleText : String = new Stringnull("Version 0.1")
- description : String = new Stringnull("Description")
- menuItems : Vector<String> = new Vector<String>()
- achievementItems : Vector<String> = new Vector<String>()
- stringMaps : Map<String,Integer> = new HashMap<String,Integer>()
- menuItemSelected : int = 0
- achievementItemSelected : int = 1
~ achievements : ArrayList<String> = null
+ AchievementState()
+ init(gc : GameContainer, sbg : StateBasedGame) : void
+ render(gc : GameContainer, sbg : StateBasedGame, g : Graphics) : void
+ update(gc : GameContainer, sbg : StateBasedGame, delta : int) : void
+ getID() : int
+ keyPressed(key : int, c : char) : void
+ keyReleased(key : int, c : char) : void
+ inputEnded() : void
+ inputStarted() : void
+ isAcceptingInput() : boolean
+ setInput(input : Input) : void
+ getFont() : Font
+ setFont(newFont : Font) : void
+ getMenuItems() : Vector<String>
+ setMenuItems(menu : Vector<String>) : void
+ getStringMaps() : Map<String,Integer>
+ setStringMaps(strings : Map<String,Integer>) : void
Portal2D
+ LOADSTATE : int = 0
+ MAINMENUSTATE : int = 1
+ ACHIEVEMENTSTATE : int = 2
+ HIGHSCORESTATE : int = 3
+ OPTIONSTATE : int = 4
+ GAMESTATE : int = 5
+ PAUSESTATE : int = 6
+ screenwidth : int = 1024
+ screenheight : int = 768
+ fullscreen : boolean
+ Portal2D()
+ initStatesList(gc : GameContainer) : void
+ main(args : String[]) : void
+ setResolution(width : int, height : int, key : boolean) : void
+ setFullscreen() : void
- app
AppGameContainer
StateBasedGame
LoadingState
- StateId : int = Portal2D.LOADSTATE
- finisheJloading : boolean = false
- levelToload : int = 0
+ LoadingState()
+ init(gc : GameContainer, sbg : StateBasedGame) : void
+ render(gc : GameContainer, sbg : StateBasedGame, g : Graphics) : void
+ update(gc : GameContainer, sbg : StateBasedGame, delta : int) : void
+ loadLevel(sbg : StateBasedGame, levelId : int) : void
+ loadNextLevel(sbg : StateBasedGame) : void
+ getID() : int

Figure 8

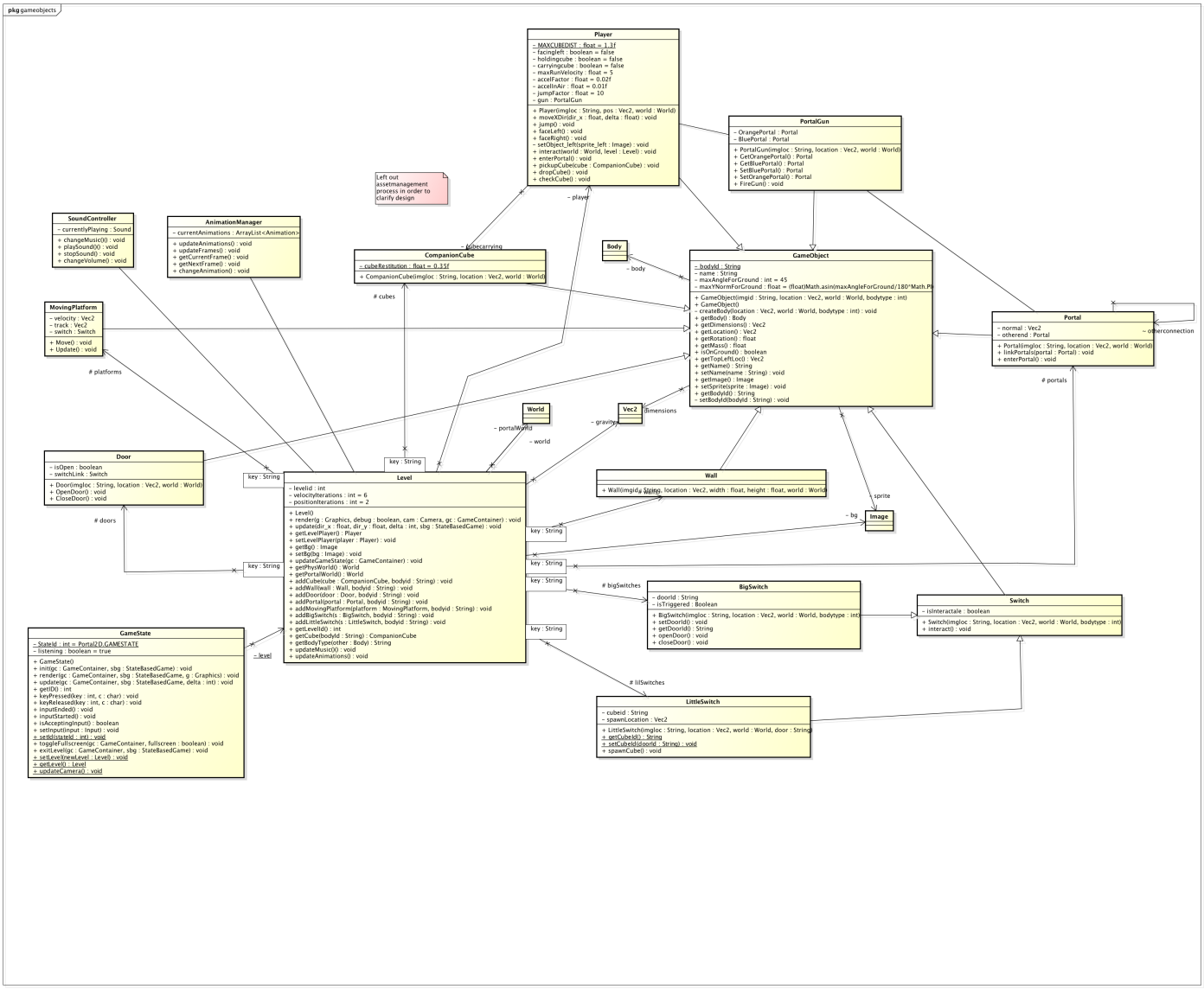
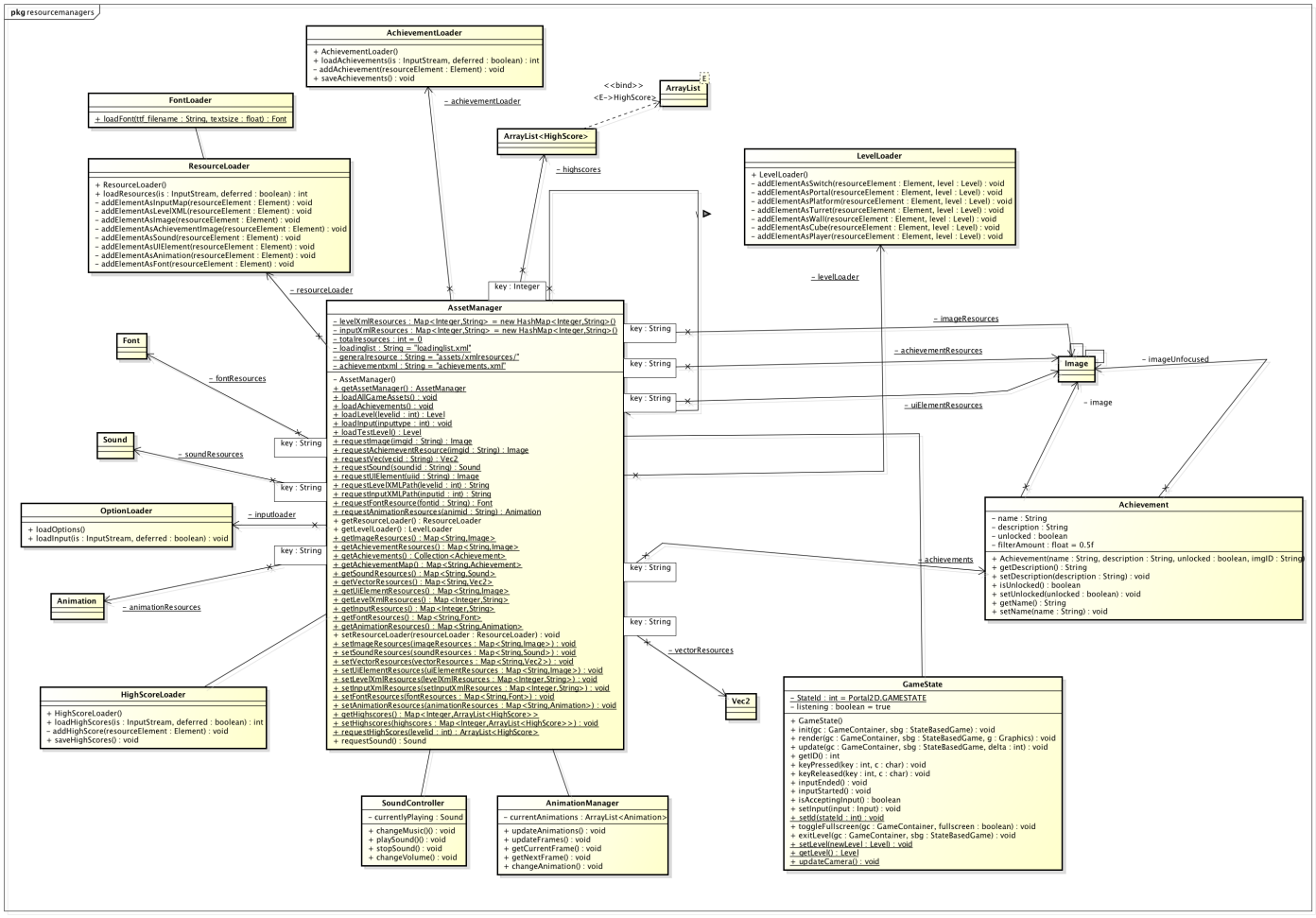

pkg gameobjects
Player
- MAXCUBEDIST : float = 1.1f
- facingLeft : boolean = false
- holdingcube : boolean = false
- carryingcube : boolean = false
- maxRunVelocity : float = 5
- accelFactor : float = 0.02f
- accelInAir : float = 0.01f
- jumpFactor : float = 10
- gun : PortalGun
+ Player(imgloc : String, pos : Vec2, world : World)
+ moveXDir(dir_x : float, delta : float) : void
+ jump() : void
+ faceLeft() : void
+ faceRight() : void
- setObject_left(sprite_left : Image) : void
+ interact(world : World, level : Level) : void
+ enterPortal() : void
+ pickupCube(cube : CompanionCube) : void
+ dropCube() : void
+ checkCube() : void
PortalGun
- OrangePortal : Portal
- BluePortal : Portal
+ PortalGun(imgloc : String, location : Vec2, world : World)
+ GetOrangePortal() : Portal
+ GetBluePortal() : Portal
+ SetBluePortal() : Portal
+ SetOrangePortal() : Portal
+ FireGun() : void
Left out assetmanagement process in order to clarify design
- player
- cubecarrying
SoundController
- currentlyPlaying : Sound
+ changeMusic(i) : void
+ playSound(i) : void
+ stopSound() : void
+ changeVolume() : void
AnimationManager
- currentAnimations : ArrayList<Animation>
+ updateAnimations() : void
+ updateFrames() : void
+ getCurrentFrame() : void
+ getNextFrame() : void
+ changeAnimation() : void
CompanionCube
- cubeRestitution : float = 0.35f
+ CompanionCube(imgloc : String, location : Vec2, world : World)
# cubes
Body
- body
GameObject
- bodyId : String
- name : String
- maxAngleForGround : int = 45
- maxYNormForGround : float = (float)Math.asin(maxAngleForGround/180*Math.PI)
+ GameObject(imgid : String, location : Vec2, world : World, bodytype : int)
+ GameObject()
- createBody(location : Vec2, world : World, bodytype : int) : void
+ getBody() : Body
+ getDimensions() : Vec2
+ getLocation() : Vec2
+ getRotation() : float
+ getMass() : float
+ isOnGround() : boolean
+ getTopLeftLoc() : Vec2
+ getName() : String
+ setName(name : String) : void
+ getImage() : Image
+ setSprite(sprite : Image) : void
+ getBodyId() : String
- setBodyId(bodyId : String) : void
Portal
- normal : Vec2
- otherend : Portal
- otherconnection
+ Portal(imgloc : String, location : Vec2, world : World)
+ linkPortals(portal : Portal) : void
+ enterPortal() : void
# portals
MovingPlatform
- velocity : Vec2
- track : Vec2
- switch : Switch
+ Move() : void
+ Update() : void
# platforms
World
- portalWorld
- world
Vec2
- gravity
- dimensions
Door
- isOpen : boolean
- switchLink : Switch
+ Door(imgloc : String, location : Vec2, world : World)
+ OpenDoor() : void
+ CloseDoor() : void
# doors
key : String
Level
- levelId : int
- velocityIterations : int = 6
- positionIterations : int = 2
+ Level()
+ render(g : Graphics, debug : boolean, cam : Camera, gc : GameContainer) : void
+ update(dir_x : float, dir_y : float, delta : int, sbg : StateBasedGame) : void
+ getLevelPlayer() : Player
+ setLevelPlayer(player : Player) : void
+ getBg() : Image
+ setBg(bg : Image) : void
+ updateGameState(gc : GameContainer) : void
+ getPhysWorld() : World
+ getPortalWorld() : World
+ addCube(cube : CompanionCube, bodyId : String) : void
+ addWall(wall : Wall, bodyId : String) : void
+ addDoor(door : Door, bodyId : String) : void
+ addPortal(portal : Portal, bodyId : String) : void
+ addMovingPlatform(platform : MovingPlatform, bodyId : String) : void
+ addBigSwitch(s : BigSwitch, bodyId : String) : void
+ addLittleSwitch(s : LittleSwitch, bodyId : String) : void
+ getLevelId() : int
+ getCube(bodyId : String) : CompanionCube
+ getBodyType(other : Body) : String
+ updateMusic() : void
+ updateAnimations() : void
Wall
+ Wall(imgid : String, location : Vec2, width : float, height : float, world : World)
sprite
Image
- bg
bigSwitches
BigSwitch
- doorId : String
- isTriggered : Boolean
+ BigSwitch(imgloc : String, location : Vec2, world : World, bodytype : int)
+ setDoorId() : void
+ getDoorId() : String
+ openDoor() : void
+ closeDoor() : void
Switch
- isInteractale : boolean
+ Switch(imgloc : String, location : Vec2, world : World, bodytype : int)
+ interact() : void
# lilSwitches
LittleSwitch
- cubeId : String
- spawnLocation : Vec2
+ LittleSwitch(imgloc : String, location : Vec2, world : World, door : String)
+ getCubeId() : String
+ setCubeId(doorId : String) : void
+ spawnCube() : void
GameState
- StateId : int = Portal2D.GAMESTATE
- listening : boolean = true
+ GameState()
+ init(gc : GameContainer, sbg : StateBasedGame) : void
+ render(gc : GameContainer, sbg : StateBasedGame, g : Graphics) : void
+ update(gc : GameContainer, sbg : StateBasedGame, delta : int) : void
+ getID() : int
+ keyPressed(key : int, c : char) : void
+ keyReleased(key : int, c : char) : void
+ inputEnded() : void
+ inputStarted() : void
+ isAcceptingInput() : boolean
+ setInputInput : Input) : void
+ setId(stateId : int) : void
+ toggleFullscreen(gc : GameContainer, fullscreen : boolean) : void
+ exitLevel(gc : GameContainer, sbg : StateBasedGame) : void
+ setLevel(newLevel : Level) : void
+ getLevel() : Level
+ updateCamera() : void
- level

Figure 9


pkg resourcemanagers
AchievementLoader
+ AchievementLoader()
+ loadAchievements(is : InputStream, deferred : boolean) : int
- addAchievement(resourceElement : Element) : void
+ saveAchievements() : void
- achievementLoader
<<bind>>
<E->HighScore>
ArrayList
E
FontLoader
+ loadFont(ttf_filename : String, textsize : float) : Font
ArrayList<HighScore>
- highscores
LevelLoader
+ LevelLoader()
- addElementAsSwitch(resourceElement : Element, level : Level) : void
- addElementAsPortal(resourceElement : Element, level : Level) : void
- addElementAsPlatform(resourceElement : Element, level : Level) : void
- addElementAsTurret(resourceElement : Element, level : Level) : void
- addElementAsWall(resourceElement : Element, level : Level) : void
- addElementAsCube(resourceElement : Element, level : Level) : void
- addElementAsPlayer(resourceElement : Element, level : Level) : void
ResourceLoader
+ ResourceLoader()
+ loadResourceIs : InputStream, deferred : boolean) : int
- addElementAsInputMap(resourceElement : Element) : void
- addElementAsLevelXML(resourceElement : Element) : void
- addElementAsImage(resourceElement : Element) : void
- addElementAsAchievementImage(resourceElement : Element) : void
- addElementAsSound(resourceElement : Element) : void
- addElementAsUIElement(resourceElement : Element) : void
- addElementAsAnimation(resourceElement : Element) : void
- addElementAsFont(resourceElement : Element) : void
- resourceLoader
key : Integer
- levelLoader
AssetManager
- levelXmlResources : Map<Integer,String> = new HashMap<Integer,String>()
- inputXmlResources : Map<Integer,String> = new HashMap<Integer,String>()
- totalresources : int = 0
- loadinglist : String = "loadinglist.xml"
- generalresource : String = "assets/xmlresources/"
- achievementxml : String = "achievements.xml"
- AssetManager()
+ getAssetManager() : AssetManager
+ loadAllGameAssets() : void
+ loadAchievements() : void
+ loadLevel(levelId : int) : Level
+ loadInputUI(inputtype : int) : void
+ loadTestLevel() : Level
+ requestImage(imgid : String) : Image
+ requestAchievementResource(imgid : String) : Image
+ requestVec(vecid : String) : Vec2
+ requestSound(soundId : String) : Sound
+ requestUIElement(uiid : String) : Image
+ requestLevelXMLPath(levelid : int) : String
+ requestInputXMLPath(inputid : int) : String
+ requestFontResource(fontid : String) : Font
+ requestAnimationResources(animid : String) : Animation
+ getResourceLoader() : ResourceLoader
+ getLevelLoader() : LevelLoader
+ getImageResources() : Map<String,Image>
+ getAchievementResources() : Map<String,Image>
+ getAchievements() : Collection<Achievement>
+ getAchievementMap() : Map<String,Achievement>
+ getSoundResources() : Map<String,Sound>
+ getVectorResources() : Map<String, Vec2>
+ getUIElementResources() : Map<String,Image>
+ getLevelXmlResources() : Map<Integer,String>
+ getInputResources() : Map<Integer,String>
+ getFontResources() : Map<String,Font>
+ getAnimationResources() : Map<String,Animation>
+ setResourceLoader(resourceLoader : ResourceLoader) : void
+ setImageResources(imageResources : Map<String,Image>) : void
+ setSoundResources(soundResources : Map<String,Sound>) : void
+ setVectorResources(vectorResources : Map<String,Vec2>) : void
+ setUIElementResources(uiElementResources : Map<String,Image>) : void
+ setLevelXmlResources(levelXmlResources : Map<Integer,String>) : void
+ setInputXmlResources(setInputXmlResources : Map<Integer,String>) : void
+ setFontResources(fontResources : Map<String,Font>) : void
+ setAnimationResources(animationResources : Map<String,Animation>) : void
+ getHighscores() : Map<Integer,ArrayList<HighScore>>
+ setHighscores(highscores : Map<Integer,ArrayList<HighScore>>) : void
+ requestHighScores(levelId : int) : ArrayList<HighScore>
+ requestSound() : Sound
key : String
- imageResources
- achievementResources
- uiElementResources
Image
- imageUnfocused
- image
Font
- fontResources
Sound
- soundResources
OptionLoader
+ loadOptions()
+ loadInput(is : InputStream, deferred : boolean) : void
- inputloader
Achievement
- name : String
- description : String
- unlocked : boolean
- filterAmount : float = 0.5f
+ Achievement(name : String, description : String, unlocked : boolean, imgID : String)
+ getDescription() : String
+ setDescription(description : String) : void
+ isUnlocked() : boolean
+ setUnlocked(unlocked : boolean) : void
+ getName() : String
+ setName(name : String) : void
- achievements
Animation
- animationResources
- vectorResources
Vec2
HighScoreLoader
+ HighScoreLoader()
+ loadHighScores(is : InputStream, deferred : boolean) : int
- addHighScore(resourceElement : Element) : void
+ saveHighScores() : void
GameState
- StateId : int = Portal2D.GAMESTATE
- listening : boolean = true
+ GameState()
+ initGc : GameContainer, sbg : StateBasedGame) : void
+ render(gc : GameContainer, sbg : StateBasedGame, g : Graphics) : void
+ update(gc : GameContainer, sbg : StateBasedGame, delta : int) : void
+ getID() : int
+ keyPressed(key : int, c : char) : void
+ keyReleased(key : int, c : char) : void
+ inputEnded() : void
+ inputStarted() : void
+ isAcceptingInput() : boolean
+ setInput(input : Input) : void
+ setId(stateId : int) : void
+ toggleFullscreen(gc : GameContainer, fullscreen : boolean) : void
+ exitLevel(gc : GameContainer, sbg : StateBasedGame) : void
+ setLevel(newLevel : Level) : void
+ getLevel() : Level
+ updateCamera() : void
SoundController
- currentlyPlaying : Sound
+ changeMusic() : void
+ playSound() : void
+ stopSound() : void
+ changeVolume() : void
AnimationManager
- currentAnimations : ArrayList<Animation>
+ updateAnimations() : void
+ updateFrames() : void
+ getCurrentFrame() : void
+ getNextFrame() : void
+ changeAnimation() : void