Architecture

Team: No

Team Members:

- Jack Burman
- Sam Churchill
- C Lloyd
- Sam Ralph
- Rebecca Wardle
- Jiacheng Wu

Previous Group

Team: Vega

Team Members:

- Chloe Wardle
- George Grasham
- Lewis McKenzie
- Matthew Rogan
- Haopeng Zhu
- Benjamin White

The architecture employs an object-oriented paradigm where objects encapsulate attributes and behaviours of entities. This was favoured over an entity component system where entities are composed of components which are used by systems to dictate interactions. This decision was based on the scope of the project. An object-oriented model provides a more rigid structure whfavoured as entities fit an inheritance model well and aspects of entities in this system do not change much at run time. Entity component systems work well for systems with many unique entities, however our system has relatively few. Not only this, but team members are more experienced using an object-oriented approach, so using this style allowed for better collaboration.

The concrete class diagram shows a much more detailed version of the abstract architecture with a few additions to better reflect the implementation. Figure 3 shows the inheritance relationships of entities which were added to reuse code. It also shows aggregation and composition relationships that represent how communication between objects was implemented. Methods, attributes and their visibility are shown to represent objects behaviour and what can be accessed by other objects. Some additional objects were created to help facilitate interactions such as the Teleport_process class which performs some of the teleporting tasks.

Figure 1.X are used to show particular processes in the system, like how abilities are applied.
Figure 2 is a abstract breakdown showing the various types of Objects and the relationships between them.
Figure 3 is a concrete breakdown generated by Intellj.

The Main Game is the main entity of our whole system, which is represented in an abstract form in Figure 2. This encompasses the whole game, and then further abstraction is shown in figures 1.2-1.7. It is implemented in our concrete representation by the GamePlay class.

Figure 1.1:

- Satisfies requirement FR_Healing
- Incorporates Healing Pod and Player from Figure 1.1
- Implemented by System, Player and ObjectContactListener from Figure 2
- This figure demonstrates the healing feature that is used in UR_Teleport, which encompasses both the teleport and the healing systems of the game.

Figure 1.2:

- Satisfies requirement UR_Sabotage, makes use of FR_Abilities
- Incorporates System Status Menu, System and Enemy from Figure 1.1
- Implemented by SystemStatusMenu, Hud, System, Enemy and EnemyManager in Figure 2 • The concrete representation is more complex due to the number of classes involved in this requirement, as it borrows from another large requirement. The abstract diagram represents the process involved in the sabotage process and an enemy using their ability upon being found in a picture format, but the concrete shows the classes involved and specifically which classes do what.

Figure 1.3:

- Satisfies requirement UR_Teleport
- Incorporates Teleport Menu, Teleporter and Player from Figure 1.1
- Implemented by Player, TeleportMenu, Teleporter, Teleport_Process, ObjectContactListener in the concrete, Player
- The idle state in the abstract is represented by ObjectContactListener in the concrete to determine when the Auber can teleport, effectively replacing two parts of the abstract diagram

Figure 1.4:

- Satisfies requirement FR_Abilities
- Incorporates Enemy and Ability from figure 1.1
- Implemented by Enemy, EnemyManger, Ability, Player and ObjectContactListener
- This requirement lends itself to the above Figure 1.3 and closely follows the abstract diagram.

Figure 1.5:

- Satisfies requirement FR_Demo
- Illustrates the logic used to determine the players action in the demo mode
- Implemented by Player and AiMovement

Figure 1.6:

- Satisfies requirement FR_Control and FR_Powers
- Illustrates the logic used to parse various user input, and the actions to take
- Implemented by Player, UserMovement, BaseAbility and Controller

Figure 1.7:

- Satisfies requirement UR_Arrest
- Incorporates Player, Enemy, Teleporter, AiMovement and TeleportMenu.
- Implemented by Player, Enemy, EnemyManager, Teleporter, Teleport_Process, TeleportMenu, ObjectContactListener and jail in Figure 2

Figures 1.1, 1.2, 1.3, 1.4, 1.5, 1.6 and 1.7 represent an abstract view of the software architecture. Figures 1.1, 1.3 1.4, 1.5 and 1.6 show state diagrams capturing the states of the system as an event occurs.

Figures 1.2, and 1.7 show interactions between entities in the system with a focus on the user. These were created with use of PlantUML. (https://plantuml.com/)

Figure 2 shows a component diagram describing how different components of the system communicate.

Figure 3 represents a concrete view of the software architecture. It shows a class diagram of the core entities, their features and relationships reflecting how the system was implemented.

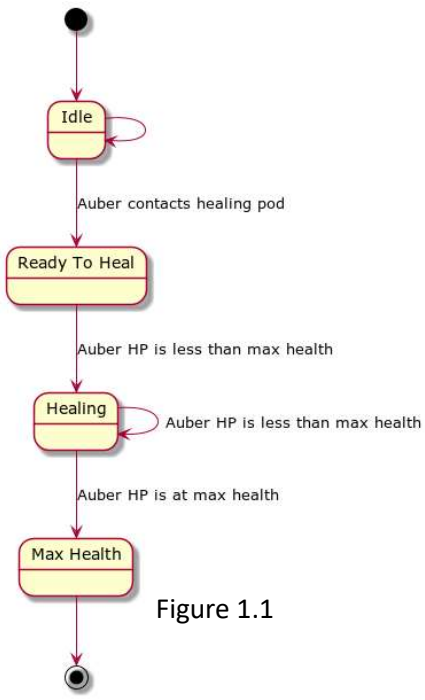

Figure 1.1

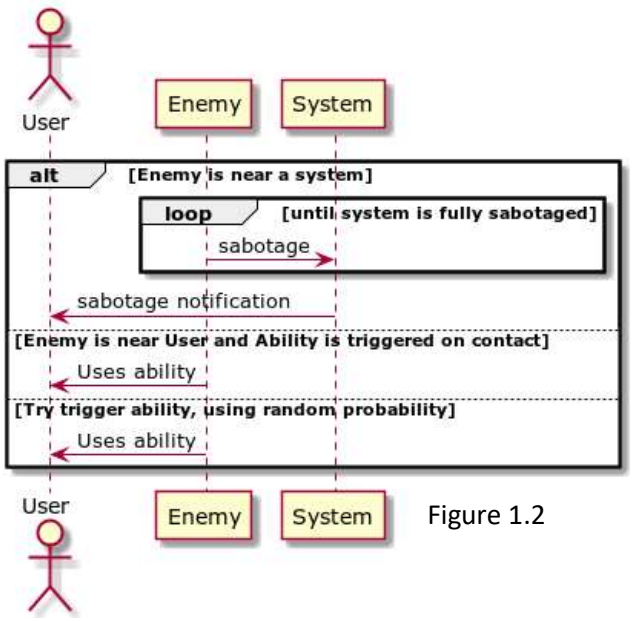

Figure 1.2

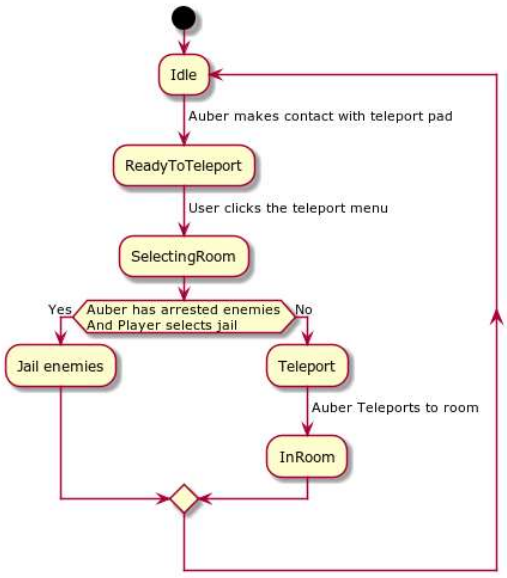

Figure 1.3

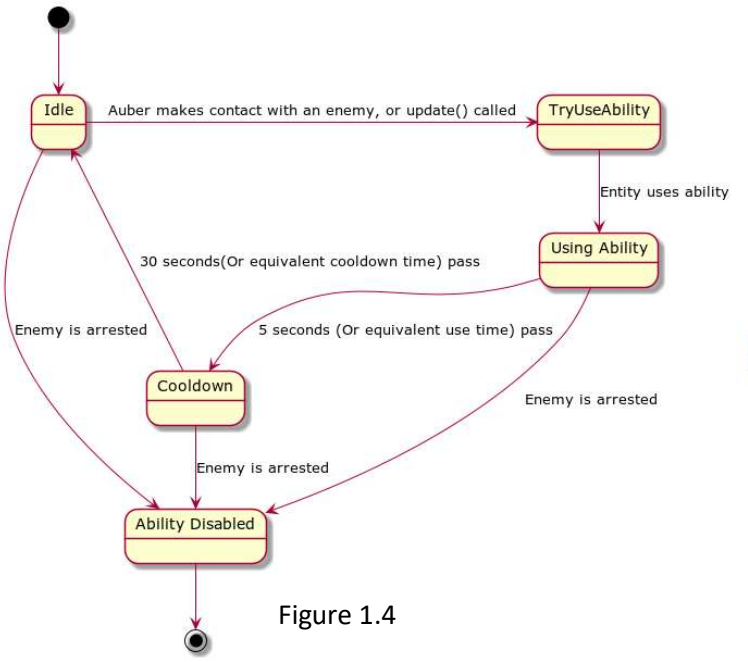

Figure 1.4

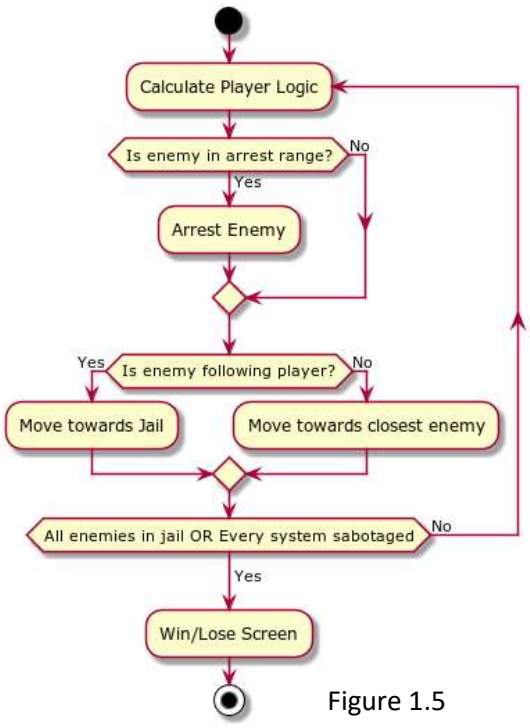

Figure 1.5

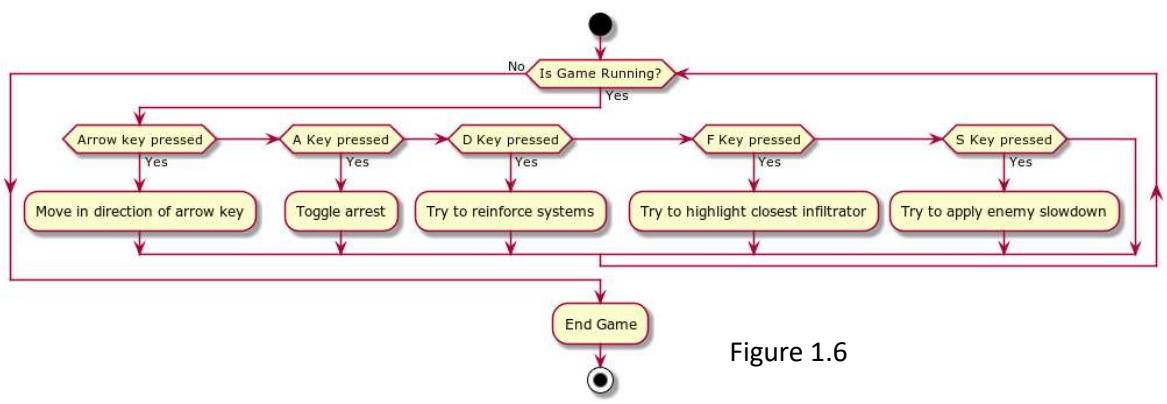

Figure 1.6

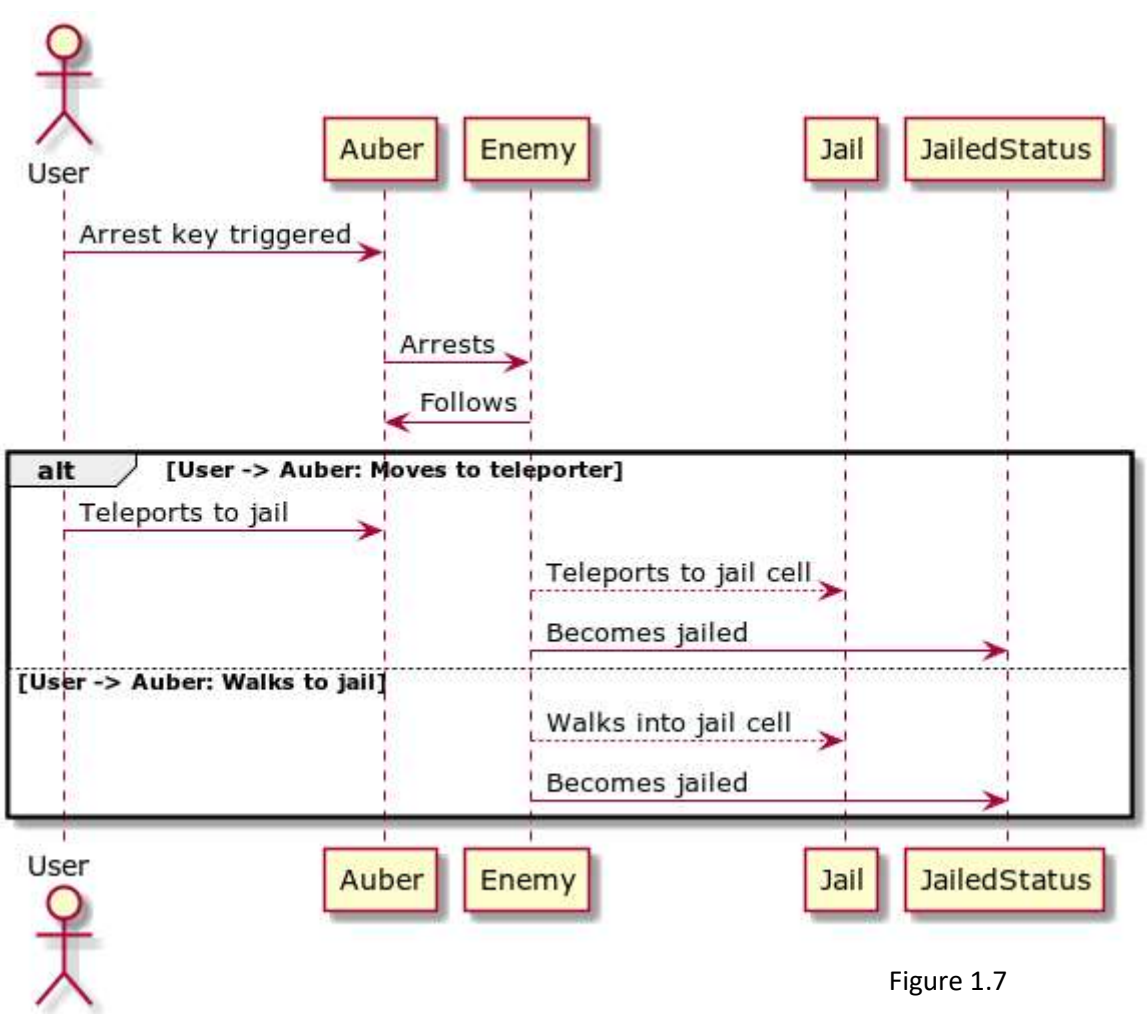

Figure 1.7

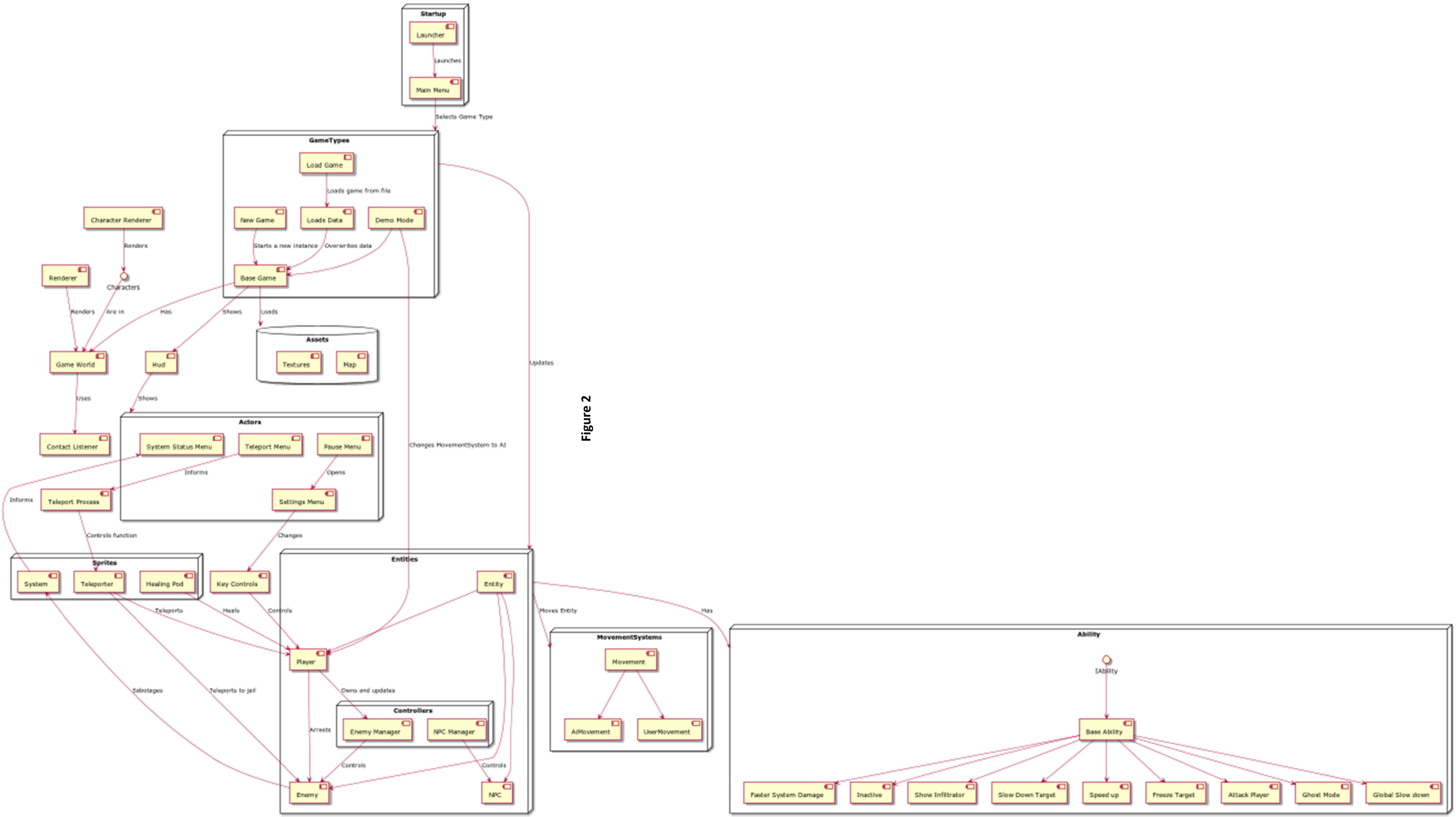
Startup
Launcher
Launches
Main Menu
Selects Game Type
GameTypes
Load Game
Loads game from file
New Game
Loads Data
Demo Mode
Starts a new instance
Overwrites data
Base Game
Character Renderer
Renders
Renderer
Characters
Renders
Are in
Has
Shows
Loads
Assets
Textures
Map
Game World
Hud
Uses
Shows
Actors
Contact Listener
System Status Menu
Teleport Menu
Pause Menu
Informs
Opens
Informs
Teleport Process
Settings Menu
Controls function
Changes
Changes MovementSystem to AI
Updates
Sprites
System
Teleporter
Healing Pod
Key Controls
Entities
Entity
Teleports
Heals
Controls
Player
Moves Entity
Has
MovementSystems
Movement
AiMovement
UserMovement
Ability
IAbility
Base Ability
Faster System Damage
Inactive
Show Infiltrator
Slow Down Target
Speed up
Freeze Target
Attack Player
Ghost Mode
Global Slow down
Sabotages
Teleports to Jail
Owns and updates
Controllers
Enemy Manager
NPC Manager
Arrests
Controls
Controls
Enemy
NPC

Figure 2

Figure 3