2. Software Architecture

2.1. Subsystem Decomposition

Three-tier architecture is chosen since it fits for our project's structure. Three-tier architecture consists three parts: presentation, functional/process logic and data storage. Diagram that shows the project's three-tier principle is given below as Figure 1.

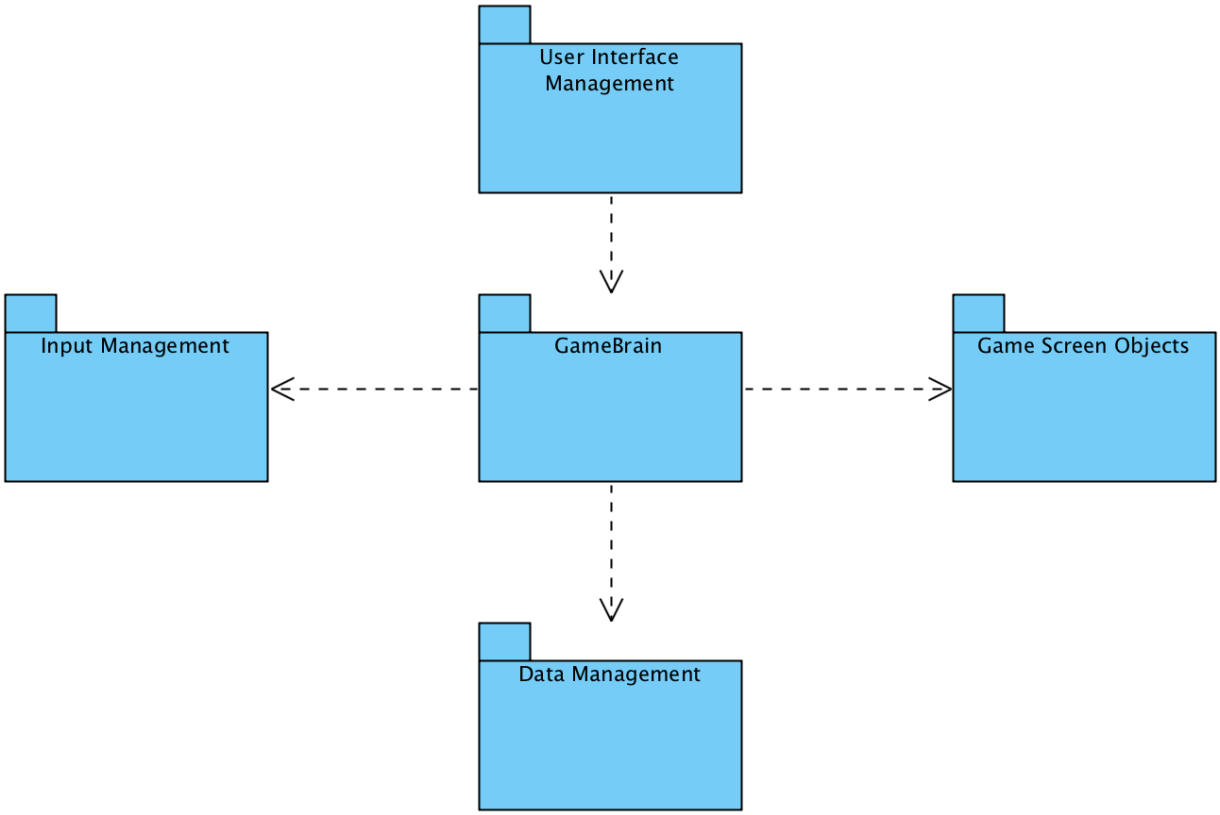

Figure 1 High Level Representation of Subsystem Decomposition.

User interface and its components are exist in the presentation tier. This tier is the first view that user faces with. Therefore, our MainMenu and all of its panels, buttons and components live at this tier. After user makes his/her actions on this tier, all of the signals get sent to the GameManager package to proceed.

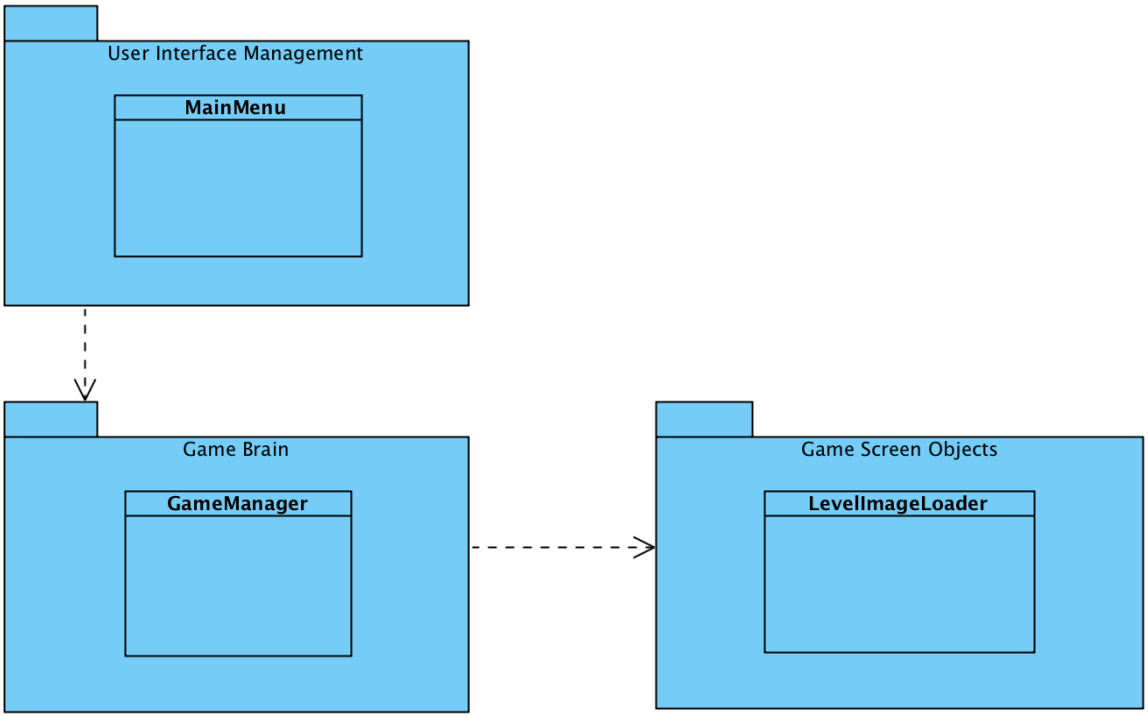

Figure 2 – Interaction between Tier 1 and Tier 2

To explain further actions, If the user clicks on the button “Play”, program will go to our “LevelPanel” to let user to select a level to play and here the logic tier is used. Then, with the help of “GameManagement” package and its helpers “LevelImageLoader”, “Camera” and “GameObject” classes, the game will be set up with its logics. Afterwards, the user will start playing the game.

The data storage layer consists of classes that keeps the data needed for objects of the game. For instance, the coordinates of the player, status of collisions and time.

2.2. Hardware / Software Mapping

Software: Our project is being developed in Java as the programming language. Therefore, Java Runtime Environment will be needed to run it.

Hardware: In terms of hardware requirements, user will use 3 buttons on keyboard to control the “Dot”, in other words to control the player to left and right, and to jump. As a computer to run the program, most of the computers can handle our game since it is an old style, not complex game.

2.3. Persistent Data Management

The only data needs to be stored in our game is “LevelsCompleted” data, which stands for keeping the levels of the game completed by the user to let user know which levels he/she has completed. This data will be stored in users’ hard drives. In addition, images of the levels will be stored in a small file that will be in the game’s files. Therefore no database will be constructed in our project to keep the track of the data.

2.4. Access Control and Security

The classes will be implemented in a way that only access to the game’s logic system and the file system will be given to the necessary class GameManager. All the variables that should not be changed at any time will be defined as constant.

2.5. Boundary Conditions

If the players dies or the time is up, the game will restart. If the player finishes a level, a small panel and two buttons on it will be shown to let user to go either next level or Main Menu.

Description of the Interactions between Classes According to the Use Cases

PlayGame: This is the main use case that user wants to start the game. When user clicks the “Play Game” button, LevelPanel that showing the levels of the game shows up on the screen. Here user will choose the level that he/she wants to play. After the level is chosen, “currentLevel” variable gets set and loadLevel() method gets called from the GameManager. Then, level gets loaded from LevelImageLoader class and drawObjects() method is called, and camera gets set on the player which is a dot and game loop starts. In this loop updateObjects() method updated the objects positions according to the user inputs to move the player around. After that, detectCollision() method checks whether the player hits any of the obstacles or gets falls down to the ground. If any collision gets detected between the player and the obstacles spike and eraser, the player will return to the beginning point to try again until the time expires. Therefore, during the game time will be checked with updateTimeRemaining() method. If there is no time left, isGameOver() method will return true and game will be over. A panel will be shown with two buttons on it to let user to choose either go back to the MainMenu or restart the level.

PauseGame: After the game starts and there will be a button “Pause” to pause the game and PauseGameMenu panel that has two buttons on it, “Resume” and “Exit”, will be shown when clicked. IsPaused variable in GameManager will be set to true when this button is clicked and will be set to false when “Resume” or “Exit” are clicked. Besides, updateTimeRemaining() method will also be stopped until the “Resume” is clicked.

ChangeVolume: This is the use case that player sets the volume using the volume bar shows up in the MainMenu class. When adjusting the volume, game will be pause on the background and it will continue when button “Exit” is clicked.

HowToPlay: HelpMenuLayout panel will be shown when button “HowToPlay” clicked. In the panel, the button information that user will need to move the player and the rules of the game will be shown.

Credits: CreditsLayout Panel will be shown when button “Credits” clicked. In the panel, the information about developers of this game will be given.

Quit: When the “Quit” button is clicked, program will save the levels that user has reached in a file that game’s file contains and terminates the program.