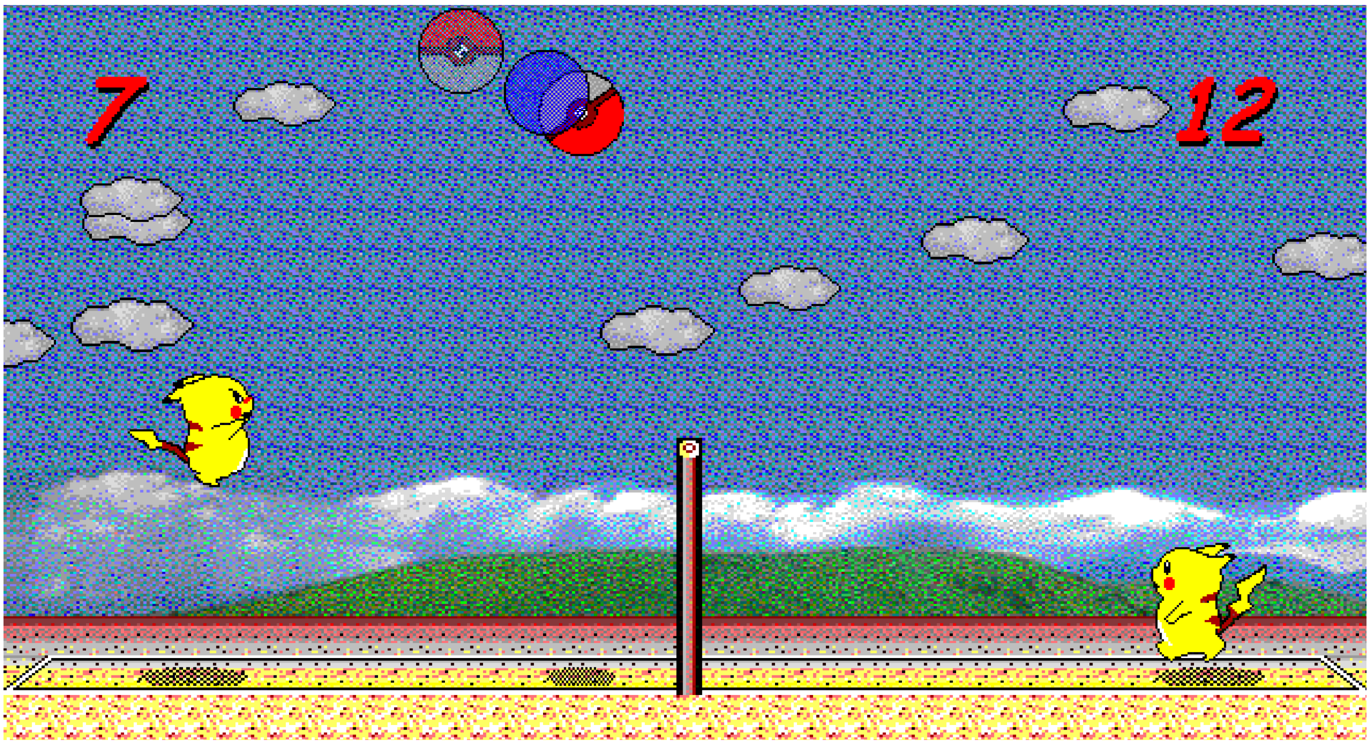
FOOP Final Project Report



Team Members & Responsibilities

- b06504025 林毓宸
  - design of pikachu
- b07501122 黃啟斌
  - design of render (background, cloud, and score)
- b07502159 黃昱翔
  - design of ball, collision, and gravity
- b07902049 謝獻沅
  - design of flow control and audio

Design

In this section, we will explain the design of each package and the relationship between them.

- background
  This package controls the background in this game.

  - **Background**
    - Extended from **Sprite**.
    - This class is like a container that background contains some instances of **Cloud**.
    - It generates 10 clouds with random velocity、shape and location to render.
  - **Cloud**
    - Extended from **VelocitySprite**
    - The instance of Cloud keep moving in the background based on its velocity.

- ball
  This package is used to build ball which contains two states: **normal** and **super**.

  - **Ball**
    - Extended from **VelocitySpite**.
    - It contains the basic information about the ball i.e., shape, finite-state-machine.
    - It provided two function **getSmash()** and **getHit()** for **Collision handler** to interact with the ball.
  - **Normal, Super**
    - Both are extended from **CyclicSequence**.
    - These two classes are used to maintain current state of the ball for both **normal hit** and **smash** respectively.
  - **SmashBallImageRender, BallImageRender**
    - Both of them implement **ImageRenderer**.
    - These classes handle how to draw the ball for both **smashed** state and **normal hit** state respectively.

- controller
  For the controller package, we reference the sample code of TA.

  - **Game**

        - Extended from **GameLoop**. Containing sprites and **World**.
        - It masters sprites' actions and counts the points of the players.
        - Provide a method for handling the moment when round is over.
    - **GameLoop**
        - Mastering the gameflow and providing **View** interface for **GameView**.
        - Handling the start and termination of the thread.
        - Set delay time for the game.

- fsm
  For the fsm package, we reference the sample code of TA.

    - **State**
        - An interface of update() and render().
    - **Sequence**
        - Useful for sequence update of image frames rendering.
        - For example, **CyclicSequence** extends **Sequence** and implement onSequenceEnd(), which goes back to the first frame after rendering every image frame.
    - **StateMachine**
        - An interface extended from **State**. Implementation can be seen in **FiniteStateMachine**.
        - **FiniteStateMachine** keeps a transitionTable to track the relationship between states and a resetTable to decide the reset state according to the current state.
        - **Pikachu** will utilize **FiniteStateMachine** to update its current state.
    - **WaitingPerFrame**
        - Useful to modify the waiting time of each frame.

- Main

    - Add the audio by path.
    - Initialize the location, initial velocity, and face for two **Pikachu** players.
    - Initialize the location and initial velocity for a **Ball** type volleyball.
    - **World** inputs **CollisionHandler**, **Gravity**, **Pikachu**, and **Ball**.
    - **Game** controls **World**. **Gameview** passes signal to **Game**.

- media
  For the media package, we reference the sample code of TA.
  - **AudioPlayer**
    - Handling sounds in the game, provide method for adding audio file path and playing the audio.

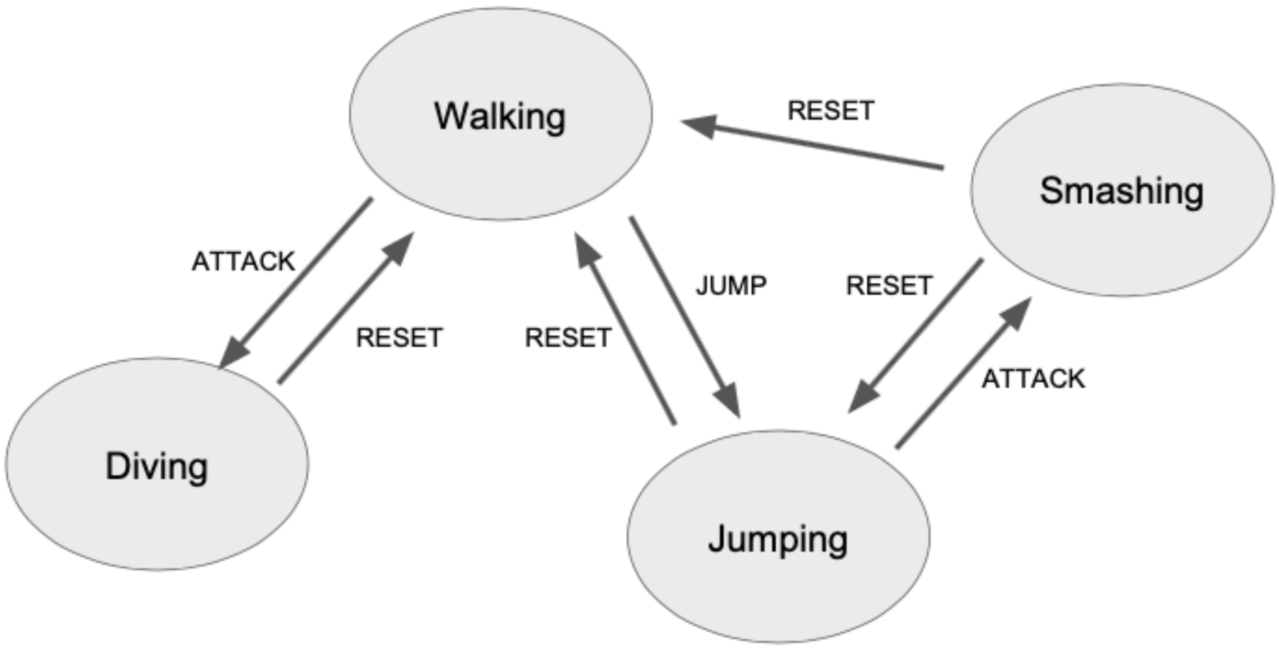
- model
  For the model package, we reference the sample code of TA on **Sprite** and **SpriteShape**. We also modify the world and create some additional sprites to fit with our scenario.
  - **VelocitySprite**
    - **Sprite** which has velocity attribute. setV() and getV() are added.
    - **Pikachu** and **Ball** can extend the **VelocitySprite** to issue the velocity.
  - **ShadowSprite**
    - A **Sprite** that render the shadow of its owner.
    - Get the x-axis location from owner.getRange() and then render shadow on the ground line.
  - **World**
    - Mastering the sprites and physic engine in the game world.
    - Use update() to call all sprites to update in every loop and render() to make every sprites render.
    - Provide move() method for every sprites to make movement and check collision with each other, as well as dealing with edge collision.
    - Define the action for reset when the round is over, and reset() will be called by **Game**.

- physic
  Here we implement some useful class to let the agent interact with each other and the world.
  - **CollisionBetweenBallAndPlayer**
    - Extended from **CollisionHandler**.
    - Used when pikachu and ball collide together. It will reset the location of the ball to prevent it from conflict to pikachu’s location. Besides, it will change the speed of the ball based on whether pikachu is smashing

the ball or not.

- **Gravity**
  - Extended from **MotionHandler**.
  - It is used to handle the gravity of all agents in the world by subtract a fixed value in a given period.

- pikachu
  For the package pikachu, we build **FiniteStateMachine** in Pikachu, and we implement four states, respectively **Walking**, **Diving**, **Jumping**, and **Smashing**.



  - **Pikachu**
    - In order to facilitate the user game experience, we introduce the velocity system in the move(), stop(), jump(), attack(). That is, the velocity only depends on the user keyrelease or user keypressed. We keep the update of the location and velocity in update() to update them for each game delay.

- scorer
  This package only contains one class **Scorer**, which is used to display the scores of two players.

  - **Scorer**
    - Extended from **Sprite**.

        - It first read the image of digit 0 to 10 from folder by **getAllDigitImagesFromFolder**.
        - At every turn, it checks the scores of two player and render them using **renderScore**.

- utils
  For the package utils, we reference the sample code from TA for **ImageStateUtils**. **Constants** stores some constants which would be used in the game, and they can be easily modified to control the setting of the game.

- views
  For the views package, we reference the sample code of TA.
    - **GameView**
        - Extended from **JFrame**.
        - Handling keyboard events and creating **Canvas** for painting.
        - **Canvas** is extended from **JPanel** and implements **View** interface in **Gameloop**, providing rendering and painting.
        - When getting keyboard events, it will call the sprite's actions in **Game**.

- assets
    - All image and audio resource are from a github repository. see **Reference**[2] .

Advantages of this Design

- Easy for extension
    - Based on TA design, this game is easy to have more functionalities. For example, Pikachu's power skills or different roles could be included.
- Free parameters for adjustment
    - We use a constant class to maintain all important parameters that afftect the whole world and all agents. As a result, we can change our game style more easily by just adjusting some variable in the constant class to achieve some game mode i.e., quicker or slower game, no gravity environment, etc.
- User experience
    - At first, the movement of pikachu is based on the current state of pikachu.

But this will issue some of the problems.
    - For example, if pikachu is smashing, we cannot interrupt the smashing state to the sky walking state. That is, even if user keeps pressing left and right when pikachu is smashing, pikachu will not move since it is in the state of smashing.
    - Thus, we improve our velocity update once user presses the left or right bottom, and we cancel the velocity if the user releases the left or right bottom. After the modification, the game experience becomes better.

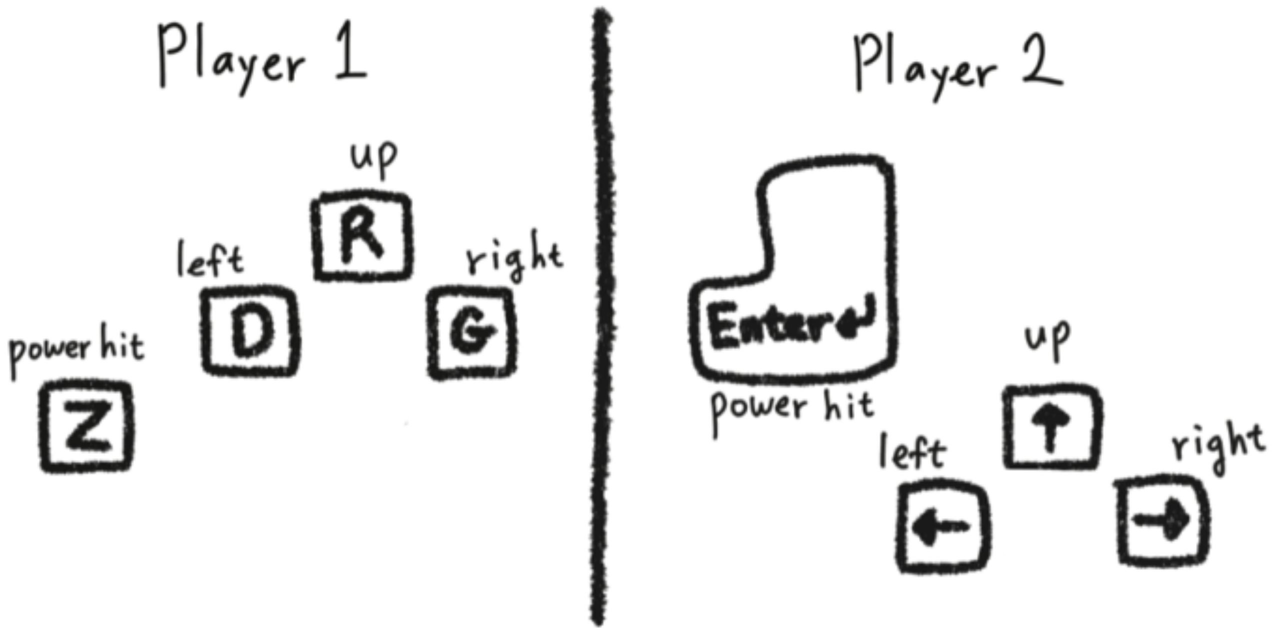
Disadvantages of this Design

- Render
    - As TA said, the class “java.awt.Graphics” is directly coupling with the model, making it difficult to modify for other renderers.
- Velocity system
    - As we mentioned in the advantage section, we modify velocity update as keyboard trigger.
    - However, the design of velocity update couples with the method of **Pikachu**, making it look complex or noodle-oriented in those source code.
    - Design of velocity system is preferable to make the design reasonable.

Packages

- Java Swing

How to play



Reference

[1] https://github.com/Johnny850807/Java-Game-Programming-with-FSM-and-MVC (https://github.com/Johnny850807/Java-Game-Programming-with-FSM-and-MVC)
[2] https://github.com/gorisanson/pikachu-volleyball (https://github.com/gorisanson/pikachu-volleyball)