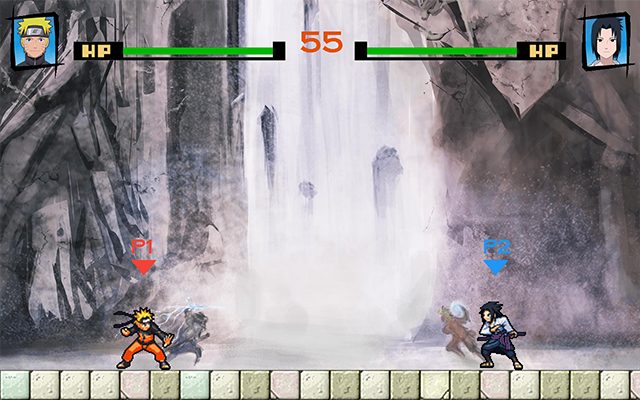
**Stage 2 Documentation and marking sheet**

**Full name:** Zhi Bohao

**Username:** Psybz2

**Student id:** 4285624

**Screenshot: **

**Summary of how you have met each requirement.**

**Attractive background: (Correctly drawn, animated or scrolling, complex behaviour : 3 marks)**

I use different images to draw backgrounds for each stage. When stage has changed, the animation effect will be used to switch the background image. I create a class “BackgroundControl” to control the switch of the background. Using this class can let me apply various kinds of animation effects, which may be complex, to each stage switch.

At the teachStage, when character has gone to the edge of the screen, the background will scroll.

3 marks.

**Multiple states/Stages: (2 states, 3 different states/levels, polymorphic state classes: 3 marks)**

There are 8 states in my game, stateMenu, stateSelect, stateBattle, statePaused, stateTransition, stateTeach, stateOver, and stateRecord. When stage has changed, the background will be redrawn correctly.

2 marks.

**User Controlled Object: (keyboard/mouse, keyboard&mouse+feature: 2 marks)**

User can use keyboard and mouse to control the character. Using keyboard to move, defence, attack, jump or teleport. Clicking the mouse can make character to show a victory posture. When character is attacked, or doing some specific actions, user cannot take control of him.

2 marks.

**Automated moving objects: (basic, 2 different ones, good class hierarchy: 3 marks)**

There are 4 automated moving objects, BirdObject, KakashiObject, SasukeAI and NameBarObject. CharacterObject is a subclass of DisplayableObject. And BirdObject, KakashiObject and SasukeAI are subclasses of CharacterObject. SasukeAI will use different action to fight user.

SasukeAI has used the behaviour BeAttacked which is in the ChatacterObject class. While KakashiObject is cannot be attacked, so this class does not use behaviour BeAttacked.

3 marks.

**Moving objects and background: (interaction, 1 mark)**

In the teachStage, after understanding the key operation. User need to control the character go to the location specified on the background. The transitions will be triggered and stage will be moved to the next.

1 mark.

**Interaction between objects: (basic/tilebased, more complex/pixel based/reliable: 2 marks)**

Each character has a defence box which includes his body. When character is in the attack action, he will have an attack box which will make damage. When the attack box is overlapping the defense box of other character, this character will be hurt. The collision detection is pixel based, and will make character act reliable and appropriately.

**C:\Users\Administrator\AppData\Local\Microsoft\Windows\INetCache\Content.Word\未标题-1.png** Green area is defense box. Red area is attack box.

2 marks.

**Tile Manager: (basic which changes, full with some complexity: 2 marks)**

I draw the tile at the bottom of the screen, the character can stand on it. When some character has done special attacks, the tile will crack. And after several seconds, the tile will be repaired automatically. Tile redraw correctly and word well.

2 marks.

**Load/save: (basic high score, more complex eg maps, state: 2 marks)**

The game can save the results of each match. At the recordStage, users can view the result of latest 8 matches. The most recent game will appear on the top.

2 marks.

**Text: (changing text over time, more complex e.g. on displayable object: 2 marks)**

Time will be shown on the top during game. If the battle is vs human, the name bar, P1 or P2, will on the top of character and moving with the character.

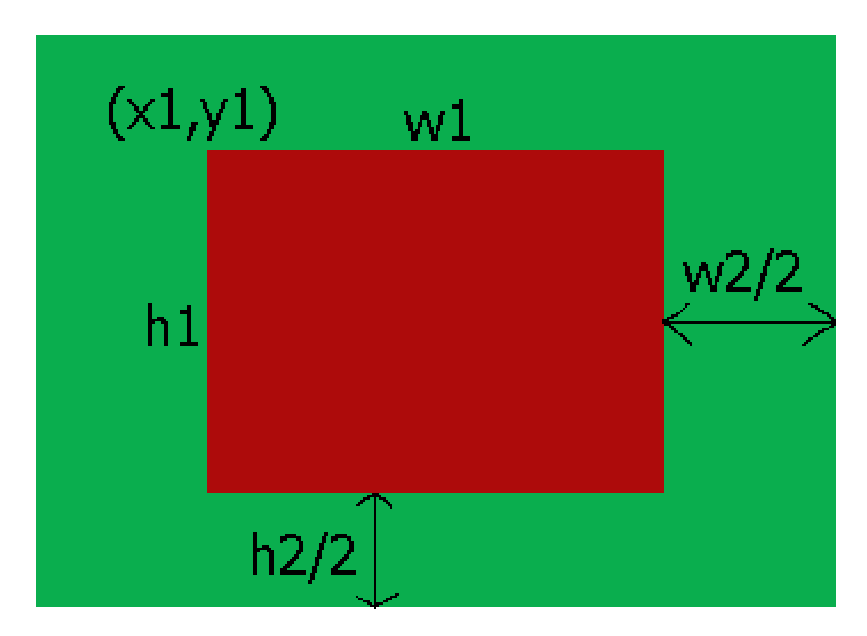
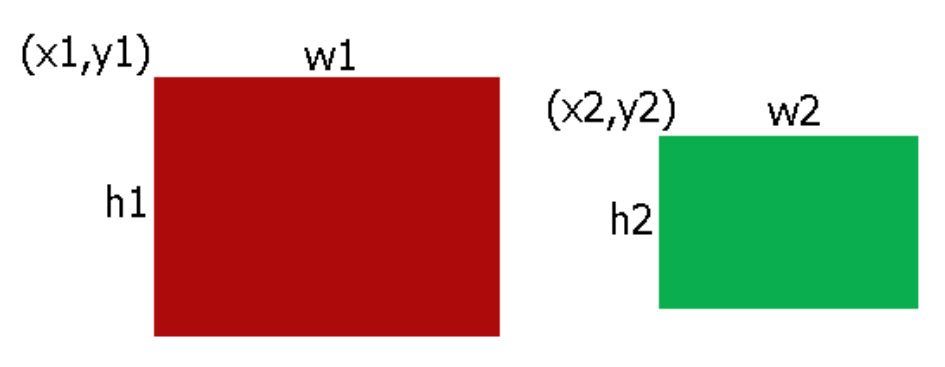
At the teachStage, the text will teach user how to play game. If the user control the character to make the right behavior, the text will change.

2 marks.

**Complex algorithm: (basic attempt, non-trivial, not simple for marker to do: 3 marks)**

AI can make different action according to distance and behaviors of character which can be controlled by user.

I have used an algorithm to check whether two rectangles are overlapping each other, which, in fact, use a little mathematical knowledge.



The centre coordinates of rectangle2:(x2 + w2/2, y2 + h2/2)

If these two rectangles are overlapping each other, the centre coordinates of retangle2 must in large rectangle.

3 marks

**Base engine container class: (not quite right, works properly, new feature added: 3 marks)**

Using vector instead of the array. All behaviour in demo is still work well without modification.

And a new feature has been added, “AddObject” can add a new object to the container directly.

3 marks.

**Wow factor: (nicely done, exceptional, wow!, extremely impressive/pay for: 4 marks)**