Summary of Gem Crush Game

WU HAOTIAN 53064039

YU YUE 53064052

In the game “Gem Crush”, our group use these classes: “Demo1”, “Gem”, “GameTimer” and “Sound”. We use these to complete the functions as listed:

1. Generate random gems (7 colors) to fill up the game board when game starts.
2. Swap any two pieces of adjacent gems.
3. Detect horizontal and vertical chains of three or more gems and eliminate them.
4. Slide the gems down to fill up holes underneath.
5. Generate random gems to fill up holes at the top.
6. Trigger chain reactions (cascades) to eliminate more gems.
7. Determine total score by counting the eliminated gems (10 points per gem).
8. Display time duration (sample code provided).
9. Produce animation for elimination of gams.
10. Play sound effects and background music (sample code provided).
11. Save and resume game.

(a). Generate the key components

For generate the gems, we first create an object of GameConsole called console and make it visible. Then we create and play a background sound using the playSound method in Sound class. Then it comes to the method called startGame. In this method, we generate 4 image buttons. And then put them on the left side of the panel. They are “start”, “save”, “load” and “exit”.

We use an 8\*8 2D array to create 64 Gem objects. These Gem objects have position, image and coordination. We rename the images of gems as 1 – 7 and randomly add images to each one of the object using Random class. When we arrange the position of the objects, we detect whether there are three or more adjacent gems in the same color form a horizontal or vertical chain. When we generate an object, we detect whether the two objects on the left and two objects ahead of it have the same image of this object. If they have, we randomly give it a new image and then check it again. Finally, we have an initialized game panel but it doesn’t display at that time.

Then we use “setBackgroung” method in GameConsole to generate the background of this game. And then generate an object called gameTimer in GameTimer class to calculate time the player use in the game and display it on the left side of the background.

We set an area for the user to click and start game. Above the area, there is the “start” image. So if the user click the start image, the game begins. If the user don’t click on this area, when they click on the area for the gems, the game won’t begin. When “start” is clicked, the timer starts to calculate and display time and the gems appear on the screen. Then you can start playing.

(b). Elimination of gems

The “getClickedPoint” method can detect where the user clicks, and we put an icon on the position to show it. We detect where the first click and second click are. According to the position, we use two functions “position\_y1 = (point.x - 240)/65” and “position\_x1 = (point.y - 40)/65” to link the x, y coordination to the index of the gem. When user clicks two gems, if they near each other, they will swap the location by swapping the pictures they have and create two new objects and have the new images of them and replace the old ones. If the two gems aren’t adjacent with each other or the two area are in the same place of one object, the highlights of them disappear and they are unclicked. You can click new two gems.

Then the program detect whether there are three or more gems having same image on a horizontal or vertical chain by the frequency of 25 times one second by using “elimination” method. In “elimination”, the program checks whether there are three or more gems having same image in a row or a column in order.

When in “checkRow” method, if the gems having same pictures are three or more, the program calculates the number of the same objects and add the scores. For the score adding, we use another method which is showed in the next part. Then we call another thread and these places having same picture will be replaced by the new gems having a picture of fire to demonstrate they are eliminated by the user. After some time delay, these new gems will disappear and be replaced by the new gems having the images of the gems above them to generate a fall. For the first row, we generate new objects having images randomly distributed.

When the “checkCol” method, if the gems having same pictures are three or more, the program calculates the number of the same objects and add the scores. Then we call another thread and these places having same picture will be replaced by the new gems having a picture of fire to demonstrate they are eliminated by the user. After some time delay, these new gems will disappear and be replaced by the new gems having the images of the gems above them to generate a fall. For the above places who don’t have the above gems, the program generates new objects and randomly distributes images to them.

The program checks the whether the gems should be update by checking whether there are three or more adjacent gems having same images and refresh the scores and time passed by 25 time per second.

(c). Other properties

If the user press the area of “Save”, the program will generate a txt file to store the current score and the current gem format. To store gem format, we use number 1 to 7 to represent the name of the images and store the number. At that time, a new game starts.

If the user press the area of “Load”, the program will extract the format and score saved last time and arrange them on the screen. The score is the same the user get last time.

If the user press the area of “Exit”, there will be a JOptionPanel appearing and it will ask you "Click YES to Exit the Game". If you click yes, the game will be closed and in the contrary, it will stay in the previous panel.

For calculating the scores, the method “scoreAdd” will be called. If you eliminate n gems, the score will add 10\*n scores to itself and refresh it.

For sound, when a gem is selected, there will be a sound called “select.wav” rings which is written in “toggleFocus” method in “Gem” class. And when the gems are eliminated, the scores add and at the same time the sound called “match” rings. When the gems fall down, the sound “fall” rings.

(d). UML Diagram

|  |
| --- |
| Demo1.java |
| -console: final GameConsole  -score: String |
| +startGame(): void  -checkLeft(Gem gem1, Gem gem2): boolean  -checkUp(Gem gem1, Gem gem2): boolean  -checkNearGems(int x1, int y1, int x2, int y2): boolean  -scoreAdd(int x): String  -elimination(Gem[][] gem): void  -checkRow(final Gem[][] gem, int j, int I, int endPlace): boolean  -checkCol(final Gem[][] gem, int j, int I, int endPlace): boolean  -checkRight(Gem[][] gem, int i, int j): int  -checkDown(Gem[][] gem, int i, int j): int  -delayRow(final int i, final int j, final int endPlace, final Gem[][] gem, int num): void  -dropRow(final int i, final int j, final int endPlace, final Gem[][] gem): void  -delayCol(int i, int j, int endPlace, final Gem[][] gem, int count): void  -dropCol(final int i, final int j, final int endPlace, final Gem[][] gem): void  -initialGem(Gem[][] gem): void |

|  |
| --- |
| Gem.java |
| +orgY: static final int  +orgX: static final int  +w: static final int  +h: static final int  -posX: int  -posY: int  -selected: boolean  -pic: Image  -focus: Image |
| +display(): void  + isAt(Point point): boolean  +getPic(): Image  + setPic(String file): void  +setPic(Image pic): void  + getPosX(): int  + getPosY(): int  + setPosX(int posX): void  + setPosY(int posY): void  + isSelected(): boolean  + setSelected(boolean selected): void  + toggleFocus(): void  + moveTo(int x, int y): void |

|  |
| --- |
| Sound.java |
| -soundPath: String |
| +playSound():void |

|  |
| --- |
| GameTimer.java |
| -initTime: long |
| +start(): void  + getTimeString(): String |

(e). Game Screenshot

