**Final Graphics Report**

**Major developments/breakthroughs(reference specific code please):**

There were a few major developments/breakthroughs. The very first major development was getting the space background to move across the screen (Version 5). In the beginning, the concept seemed really simple. I decided to create a scratch code that looked this like this:

ActionListener updateTask = new ActionListener() {

@Override

public void actionPerformed(ActionEvent evt) {

collision();

aste.update(); // update the (x, y) position

space.moveX();

repaint();

}

};

This is the same timer I used to update my asteroid’s position. I used this to update the background’s position as well. The code used to update the background looks like this:

public void moveX() {

nx -= ndx;

}

ndx is the displacement of the x coordinate. This made the background move across the screen. It took me a few minutes to realize that I could run two parallel images. So I decided to put this bit of code in my paint function:

g2d.drawImage(space.getImage(), space.getX(), space.getY(), getWidth(), getHeight(), this);

g2d.drawImage(space.getImage(), space.getsX(), space.getY(), getWidth(), getHeight(), this);

This bit of code draws two space images at once. Now the trick was to make one follow the other. To do so, I added this code right below the code above:

if (space.getX() == - 980) {

space.updateX();

} else if (space.getsX() == -980) {

space.updatesX();

}

This code updates the x position of those two parallel images. The update functions look like this:

public void updateX() {

nx = nsX + 980;

}

public void updatesX() {

nsX = nx + 980;

}

This code is specifically designed for one image to follow the other. If the first image reaches -980 x position, it will immediate restart from the right hand side following the other image that is still moving across the screen.

Moreover, a major breakthrough was made when I finally got multiple asteroids moving across the screen (Version 6). After consulting multiple sources, I came across a really good source that taught me how to create an array of images. Unfortunately, I was forced to use arraylist to optimize the efficiency/length of my code. A 2d arraylist was first created. The idea was to add a random Y coordinate and an X value of one thousand. I made the following method to incorporate this idea:

public void addAste() {

Random random = new Random();

int nRandomY = random.nextInt((500 - 0) + 1) + 0;

arlAste.add(new Asteriods(1000, nRandomY));//Add X and Y (random y)

}

This function was placed in Spacecraft class since the asteroid class’s constructor required the values of the updated X and Y coordinates from the 2d arraylist (public Asteriods(int nXupdate, int nYupdate) ). To paint these asteroids, I used the following code :

ArrayList arlAste = spCraft.getAsteroids();

for (int i = 0; i < arlAste.size(); i++) {

Asteriods asteroids = (Asteriods) arlAste.get(i);

g2d.drawImage(asteroids.getImage(), asteroids.getX(), asteroids.getY(), this);

}

It loops through the arraylist size and draws all the asteroids.

**Major Challenges/setbacks( reference specific code please):**

The biggest setback I had while working with my graphics project is the ending of my game. When the spaceship dies, an image is displayed that says “You have failed the space adventure. Press B to restart”. It was difficult to get this image to popup at the end. However, I added a Boolean variable that assess the situation. In the paint function this code was implemented:

if (bShow) {

//Draw everything else such as space background, asteroids, health bar etc.

} else {

g2d.drawImage(imgFail, 0, 0, getWidth(), getHeight(), this);

g2d.setColor(Color.cyan);

g2d.drawString("Total Time Survived: " + nDispTime +"s", 450, 250);

}

The variable bShow is changed to false if the spaceship runs out of health. This code is placed in the loop where I add/remove asteroids and check for collision:

if ((nHealth - 20) < 0) {

timerSpace.stop();

timerAste.stop();

bShow = false;

}

If the spaceship runs out of its last health, bShow is set to false so that when repaint is called, it will only draw the final display image that alerts the user about his/her failure. Moreover, I could not get the code to go back to the main screen (menu screen) after collision so the user can restart the game. I decided not to spend my time working on it since the time was limited. Some of the other challenges I faced were:

* Drawing multiple asteroids
* Collision detection with multiple asteroids
* Transitioning to “game mode” from menu screen

All those challenges were successfully solved.

**Any modifications to your planned steps:**

The initial goals sheet was filled out without me having any prior knowledge/experience in Java GUI programming. As I was programming I realized that my planned steps were very difficult for me to follow. Therefore, there were a lot of modifications to it. Here are my revised planned steps.

|  |  |
| --- | --- |
| **Versions** | **New incremental feature of each step:**  (Each step is linked to a version that will contain that planned step) |
| **Space Adventure V.1** | **A frame loaded with the main panel and option buttons** |
| **Space Adventure V.2** | **Ignoring the menu option buttons (Play, How to play, and Credits), load up the image of a spaceship, space background, and an asteroid. It will be a little challenging task to transition from menu mode to game mode. Not only that but it is time consuming to click “Play” each time I want to test my game out.** |
| **Space Adventure V.3** | **Work on making the spaceship image move around the screen** |
| **Space Adventure V.3** | **Make an asteroid move from the right side of the screen to the left side of the screen** |
| **Space Adventure V.4** | **Work on collision detection for one asteroid** |
| **Space Adventure V.5** | **Make the space background move/animate also do bounds checking** |
| **Space Adventure V.6** | **Make multiple asteroids move from east to west (towards the spaceship)** |
| **Space Adventure V.6** | **Work on the collision detection code for multiple asteroids** |
| **Space Adventure V.6** | **Work on the health bar and asteroids bar. Health bar decreases by 20 every time the spaceship collides with an asteroid. Increase the asteroids bar as the health bar decreases meaning more asteroids as the health of the spaceship decreases.** |
| **Space Adventure V.6** | **Work on a timer that increases every second the player is alive** |
| **Space Adventure V.6** | **Work on the final image displayed after the spaceship’s health reaches 0** |
| **Space Adventure V.7** | **Finally, add the menu buttons/screen again. This will be displayed when the game is launched along with a splash screen. The menu buttons are “Play”, “How to play”, and “Credits”.** |