**Asteroids**

Final Game Design Document

Austin Avery

**Overview:**

“Asteroids” will be a basic 2D animated game to demonstrate the overall skills learned during my Summer Semester in CS120.

**Premise:** The good ole’ Asteroids arcade game is the goal I am shooting for here. The user will control the flying spaceship around the screen with the corresponding arrow keys, and shoot bullets from the spaceship with the spacebar key. When bullets are shot out a pew sound will be played. When bullet hits an asteroid a breaking sound will be played. And if the spaceship is hit by an asteroid a crash sound will play. The spaceship will start in the middle of the screen and the asteroids will appear randomly from the edges of the screen and move across at random speeds. If an asteroid leaves the edge of the screen it will wrap around to the other side. You’ll start with 5 lives. If an asteroid hits the ship, you lose a life. If an asteroid is hit by a bullet, it will break into two smaller asteroids that appear from the larger one. When those two smaller asteroids are hit, they will be destroyed. There is no time limit, the scoring will be based on asteroids hit. If you shoot a large asteroid you’ll gain 50 points and if you shoot a small one you’ll gain 100 points. Once you lose all 5 lives, the score will be recorded, and the user will be given the option to play again or quit.

**Intro:** When the game begins it will show an instructional screen with two buttons. The play button will take you to the gameplay. The quit button will exit the game altogether.

**Ending:** After the player has lost all 5 lives they will be taken back to the intro screen where the last score is displayed and the options to play or quit are offered.

**State Transition Diagram:**

This is the main flow of the game and how game states will change.

A diagram of a video game

Description automatically generated

This game will consist of two states; Intro and Game. Both of these states are subclasses of simpleGE.Scene. Each time the program is run the Intro scene will be the first thing the user sees. The intro scene will have instructions and two buttons for play or quit. If the user chooses to play, they will be taken to the game scene. If the user chooses to quit, they will be taken out of the program all together.

Once the user is playing and loses 5 lives, they will be taken back to the Intro scene. This process repeats itself until the response is quit.

A screen shot of a computer

Description automatically generated**Intro Scene:**

The intro scene will be the first thing a user sees. It will provide user’s with instructions on how to play the game and two options, play or quit. The intro scene will also display the user’s previous score if they have already played. If they haven’t, previous score will be set at 0.

Visual Aspects:

* **btnPlay** – subclassed from simpleGE.Button, used to close intro scene
* **btnQuit** – subclassed from simpleGE.Button, used to close intro scene
* **lblIntro** – subclassed from simpleGE.MultiLabel, provides instructions to user
* **lblPrevScore** – subclassed from simpleGE.Label, used to display previous score

Additional Attributes:

* **prevScore** – integer of previous score passed into class initializer, displayed on **lblPrevScore**
* **response** – a string of “play” or “quit” to represent the user’s decision on **btnPlay** or **btnQuit**, string passed to main loop

**Game Scene:**

**A diagram of space craft

Description automatically generated**This is the primary Gameplay State. It will be subclassed from simpleGE.Scene.

Visual Aspects:

* **spaceship** – instance of **Spaceship** class (subclass simpleGE.Sprite)
* **asteroidLg** - instance of **AsteroidLg** class (subclass simpleGE.Sprite)
* **asteroidSm** – instance of **AsteroidSm** class (subclass simpleGE.Sprite)
* **bullet** – instance of **Bullet** class (subclass simpleGE.Sprite)
* **lblScore** – instance of **LblScore** class (subclass simpleGE.Label)
* **lblLives** – instance of **LblLives** class (subclass simpleGE.Label)

Other Attributes:

* **score** – int of score
* **lives** – int of 5 represented as **spaceship** sprite

**Assets:**

* **spaceship**(simpleGE.Sprite)
  + sprite must be facing right
  + use imageMove to rotate image of sprite
  + if keyPressed imageMove
  + use speed to set a int of speed
  + apply speed if keypress
  + colorRect(color, size) to use objects for testing purposes when I have no image
  + turnby(Angle) and forward(distance) methods may be most beneficial for spaceship motion
    - addForce seems like most realistic but complex
* **asteroidLg**(simpleGE.Sprite)
  + setImage
  + get random speed between 3-10
  + get random x,y position
  + move onto screen from position
  + wrap around screen boarders
  + breaks into **asteroidSm** when shot
* **asteroidSm**(simpleGE.Sprite)
  + setImage
  + get random speed 3-8
  + get x, y position of **asteroidLg**, move from there in random x, y
  + wrap around screen boarders
  + gets destroyed when shot
* **bullet**(simpleGE.Sprite)
  + setImage
  + get speed 5
  + shoots straight from nose of **spaceship**
  + dies at screen boarder
* **bulletSnd**(simpleGE.Sound)
  + play laser pew sound when shot from **spaceship**
* **asteroidSnd**(simpleGE.Sound)
  + play rock smash sound when **asteroidLg(Sm)** is hit by **bullet**
* **spaceshipSnd**(simpleGE.Sound)
  + play crash sound when **spaceship** is hit by **asteroidLg(Sm)**

**Milestones:**

1. get **Game** class built w/ associated background
2. get **main()** built
3. get **spaceship** sprite built, instantiate with Game class
4. get **asteroidLg** and **asteroidSm** sprites built, instantiate w/ Game class
5. get **bullet** sprite built
6. get **LblScore** class built
   1. get **lblScore** labelbuilt
7. get **LblLives** class built
   1. get **lblLives** label built
8. get **Intro** class built

NOTES:

Fix - asteroid speed, use dx and dx so speed cant be near 0

bullets will be used for collision, not bullet