Fred in a far away galaxy

Game design document

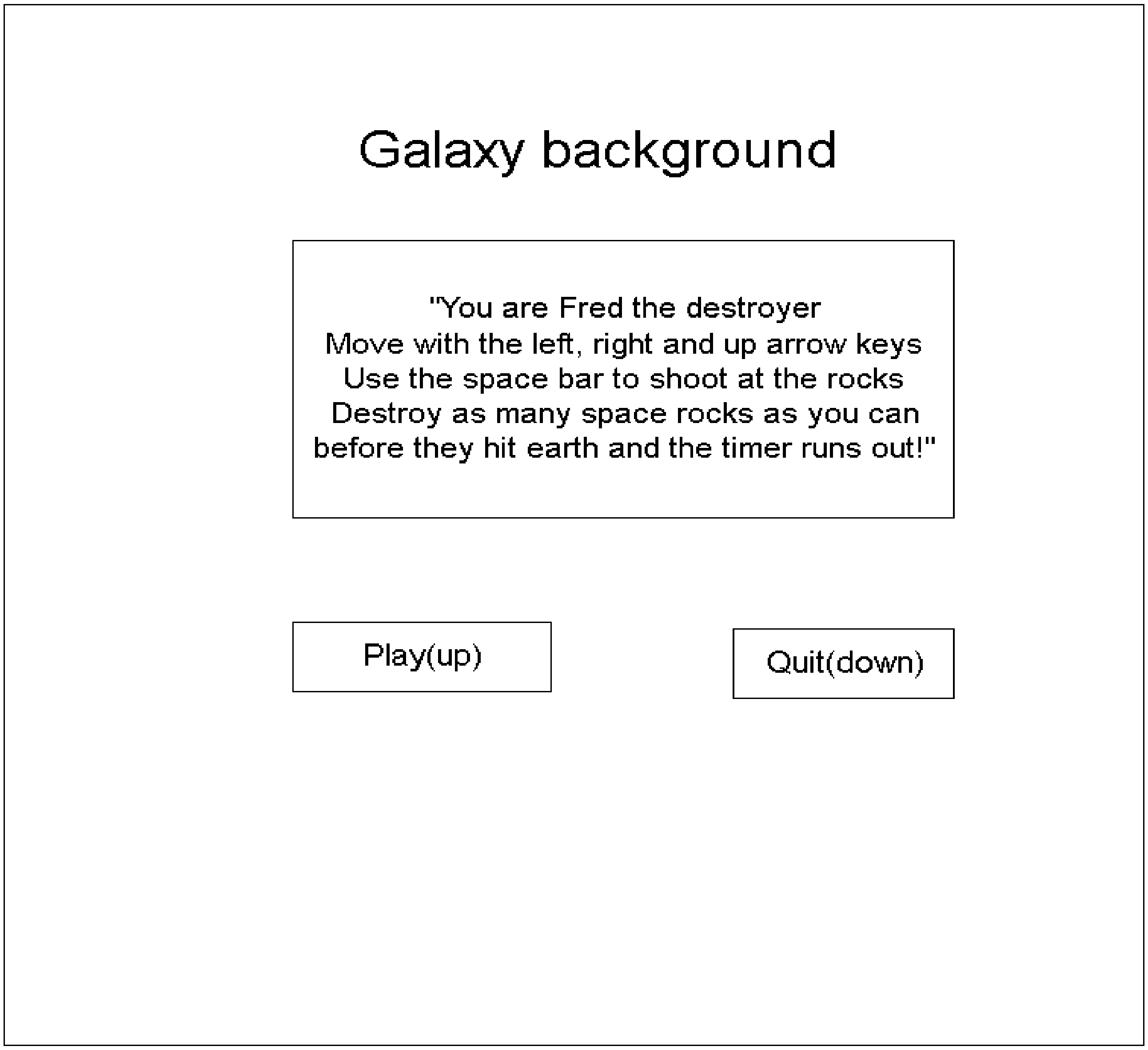
Amanda Clevenger

Overview

“Fred in a far away galaxy” is a 2D game created to allow the user to play a simple knock off version of the 1979 game “Asteroids” by using pygame and simpleGE.

This game is simple, but a fun throwback! If you loved the game “Asteroids” as a kid, then you will love my version called “Fred in a far away galaxy”. It is similar yet different in many ways. Fred is a little candy cane ghost here to destroy all the space rocks before they come crashing down to earth. He will shoot bullets but instead of the bullets blowing up the space rocks, Fred blows them up by simply running into them. Fred and the bullets are both set to a specific size but the space rocks are set at random sizes, random speeds and random angles. The user can use the left, right and up arrow to maneuver around in different directions and thrusts through the galaxy blowing up the space rocks. If the user long - holds on the up arrow, Fred will forward thrust. If the user turns Fred the opposite direction, you can counteract the direction and velocity he is moving. If the user uses a combination of all arrows, it will change his velocity and trajectory. When the game begins, there will be an instruction screen that allows the user to hit “play” to start the game and a “quit” button to end the game.

Instruction scene


Galaxy background
"You are Fred the destroyer
Move with the left, right and up arrow keys
Use the space bar to shoot at the rocks
Destroy as many space rocks as you can
before they hit earth and the timer runs out!"
Play(up)
Quit(down)

This scene has three main visual elements:
● instructions - a stock simpleGE multiLabel containing instructions for game play
● btnPlay - a stock button indicating "Play"
● btnQuit - a stock button indicating "Quit"

```
init(score):
  Set image to galaxy.jpg
  Set response to "Play"
  Create instructions MultiLabel
  Add textLines containing instructions
  Set instructions center to (320, 240)
  Set instructions size to (500,250)


  Create btnPlay
  Set text to "Play"
  Set center to (100, 400)

  Create btnQuit
  Set text to "Quit"
  Set center to (550, 400)

  Add lblInstructions, lblScore, btnQuit, and btnPlay to
sprites
```

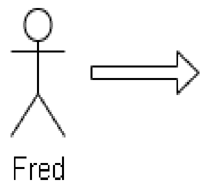
All event-handling will happen in the scene's process() method

```
process():
  If the quit button is pressed:
    Set response to "Quit"
    Stop the scene
If the play button is pressed:
    Set the response to "Play"
    Stop the scene
```

The Game Class

Right = Fred turns to the right
Left = Fred turns to the left
Up = adds force
Space = shoots bullets

Galaxy background

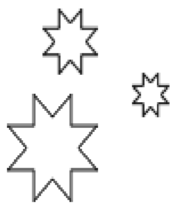


Fred the ghost

Move Fred around with the left, right and up arrows

Space rocks

Little space rocks flying around the galaxy at random speed and random sizes



If Fred collides with space rocks, they explode

Initializer will create all the needed components:

```
Init:
  Set image to galaxy.jpg
  Create timer
  Set timer's total time to 30
  Set score to zero
  Initialize sndExplosion to explosion sound effect
  Create instance of LblScore
  Create instance of LblTime
  Add dude, bullet,rock, lblScore, lblTime to sprites
```

All event-handling will occur in the scene's process() method:

```
Process:
For each rock in the rocks list:
  If that rock collides with dude:
    Play the explosion collision sound (sndExplosion)
    Reset that rock
    Add one to the score

Update lblScore to indicate the new score
Update lblTimer with the current time left
If the time left is less than zero:
  (for now) print the score to the console
  Stop the game
```

Components of the Game class

Each of the visual elements of the Game class is an extension of a simpleGE element.

Dude

Dude is a subClass of simpleGE.Sprite
Angle is adjusted with the “left” and “right” arrow key at +=5, -=5
Force is created with the “up” arrow key at 0.2
self.scene.bullet.fire() is a method called to fire bullets from Dude

```
Init:
Set image to dude.png
Set angle to 0
Set moveSpeed to 0
```

Move left on left key, right on right key

```
Process:
  If left key is pressed
Angle position increases by 5 degrees
  If right key is pressed
Angle position decreases by 5 degrees
  If up key is pressed
Force is added by 0.2 velocity
  If the spacebar is pressed
A bullet is fired from Dude
```

Rocks

Rocks is a subclass of simpleGE.Sprite
Rocks are set at an angle with rotating speeds
Fall speed is random within limits
Scale size of rocks are set to (10, 40)
If rocks leave screen, reset
Dude-rock collision handled at game level
Speed attribute is set to random(0, 6)
Angle attribute is set to random(0, 360)
Rotating speed is set to random(-5, 5)

- init() - standard initialization
- reset() - custom method to set speed and position
- checkBounds() - overwrite existing checkBounds to handle bottom-of screen

```
init():
     Set image to rock.gif
     Set random sizes to (10, 40)
     Set current angles and speed of rocks
     Call reset()
reset():
```

```
          Set X and Y to random width and height
          Set scale to random sizes
          Set speed to random(0, 6)
          Set angle to random(0, 360)
          Set rotating speed to random(-5, 5)
```

Bullets

Bullets is a subclass of simpleGE.Sprite
Set image to bullet.gif
Set size to (5, 5)
Set bounds
Rest()
Create a firing method
Set position to the same coordinates as Dude
Set speed to 12
Set angle to mirror Dude
Rest()
Set position at (-100, -100)
Set speed to 0

- init() - standard initialization
- reset() - custom method to set speed and position
- checkBounds() - overwrite existing checkBounds to handle bottom-of screen

LblScore:

LblScore is a subclass of the simpleGE.Label
It is quite simple - could have been a stock instance
It simply has text and center, no events

```
init():
Set text to "Score: 0"
Set center to (100, 30)
```

LblTime:

LblTime is also a simple subclass of simpleGE.Label
Again, only text and center, no events

```
init():
Set text to "TimeLeft: 30"
Set center to (500, 30)
```

The main() function

The main function will manage the high-level state transition between intro and play states.
It is a very standard main loop, containing four variables:

- instructions - an instance of the Instructions class
- game - an instance of the Game class
- keepGoing - the classic Boolean sentry
- score - the current score

Psuedocode for main

```
main():Set keepGoing to true
Set score to zero
While keepGoing is true:
  Create an instance of Instructions -> instructions
  Pass the current score to instructions as a parameter
  Start instructions
  When instructions ends,
  If instructions.response is "Play":
    Create an instance of Game -> game
    Start game
    When game is over, copy game.score to score
  If instructions.response is anything but "Play":
    Set keepGoing to False, which will exit the game
```

Asset Plan

Galaxy.jpg

https://unsplash.com/s/photos/bright-star

Dude.png

https://openclipart.org/detail/12550/game-baddie-candy

Rock.gif

https://opengameart.org/

Bullet.gif

https://opengameart.org/