AP Computer Science Final Project - README Template

Downfall

Authors: Kevin Ren, Aviral Vaidya
Revision: May 6, 2022

Introduction:
[In a few paragraphs totaling about ½ page, introduce the high-level concept of your program. What this looks like depends a lot on what type of thing you are making. An introduction for an *application* will look different than one for a *game*. In general, your introduction should address questions like these:
What does your program do?
What problem does it solve? Why did you write it?
What is the story?
What are the rules? What is the goal?
Who would want to use your program?
What are the primary features of your program?]

We propose a bi-dimensional, platformer game written in the programming language known as Java. Utilizing frameworks such as our very own Shapes library, and the processing library written for Java, our very own vision came to fruition.

Downfall is a perpetually continuous platformer in which you manually translate your character through the vertical and horizontal planes downwards through the sky within a constant gravitational field. The primary objective of this game is to survive as long as possible by achieving the maximum possible vertical displacement as the character falls through the sky.

The means to execute this is to land on platforms and to shoot enemies who will make attempts on your life, also by shooting you. Upon failing to steer onto a platform, the client will fall out of the sky and their turn will be terminated. As gravity accelerates you downward, the client will find themselves attempting to collide with platforms to temporarily cease your fall through careful maneuvering and tactical utilization of the arrow keys to navigate their character downward to land on such platforms. These platforms are both flat and angled, and falling on a platform causes you to rebound off along a perpendicular trajectory to the platform in an inelastic collision.

Instructions:
[Explain how to use the program. This needs to be **specific**:
Which keyboard keys will do what?
Where will you need to click?
Will you have menus that need to be navigated? What will they look like?
Do actions need to be taken in a certain order?]

The game will implement a standard menu interface for level starting or quitting, with buttons activated by mouse clicks. Upon launch, there will be a start menu with a start level button and quit button along with a help menu displaying the game controls.
In the game, you will use the left and right arrow keys to move the character left and right, respectively. You can also use WASD and press space to initiate a double jump. You can fire to the left and right at enemies using the Q and E keys.

Upon dying, you will be directed back to the menu described above with a death message.

Features List (THE ONLY SECTION THAT CANNOT CHANGE LATER):

Must-have Features:

[These are features that we agree you will *definitely* have by the project due date. A good final project would have all of these completed. At least 5 are required. Each feature should be fully described (at least a few full sentences for each)]

- Main menu at the beginning
- Platforms that move from side to side
- Enemies which loosely follow your path and kill you upon collision
- Shoot enemies with projectiles
- Angular platforms (when you hit it you bounce off at an angle)
- Screen wrap around (when you go off the edge of the screen you appear on the other side)
- Menu when you die

Want-to-have Features:

[These are features that you would like to have by the project due date, but you're unsure whether you'll hit all of them. A good final project would have perhaps half of these completed. At least 5 are required. Again, fully describe each.]

- Enemies which loosely follow your path and shoot projectiles at you
- Multiplayer (play against an opponent to see who lasts for a longer time)
- Platforms that disappear after one use
- Collectible power-ups which enhance your attacks (more projectiles, faster fire rate, etc)
- Collectible defenses which give you temporary invincibility

Stretch Features:

[These are features that we agree a *fully complete version of this program would have, but that you probably will not have time to implement*. A good final project does not necessarily need to have any of these completed at all. At least 3 are required. Again, fully describe each.]

- Varying wind (x acceleration) with animation
- Parallax effect with background
- Chat in multiplayer mode

Class List:

[This section lists the Java classes that make up the program and very briefly describes what each represents. It's totally fine to put this section in list format and not to use full sentences.]

Player.java
Represents your character which bounces on the platforms

Enemy.java
Represents the enemies

Platform.java
Represents the platforms which you land on (might have a 2nd constructor for angular as well)

Projectile.java
The projectiles launched which kill whatever it touches

Powerup.java
Power-ups that can be collected by colliding with them, they give different abilities as described above

Screen.java
The specific screen which the game is drawn on, including menu and game

DrawingSurface
This class handles the key and mouse presses that the user uses to interact with the game environment, as well as change the screen being displayed (like when you navigate through the main menu or die)

Main.java
The class with the main method. Running this class launches the program.

Game.java
Game screen

Menu.java
Menu screen seen when a player launches the game

ScreenSwitcher.java
Interface to help change screens

Sprite.java
Superclass for all objects on the screen (enemy, player, etc.)

Credits:
[Gives credit for project components. This includes both internal credit (your group members) and external credit (other people, websites, libraries). To do this:

Alpha checkpoint:

Aviral:

- Added needed jar files to project (shapes and processing)
- Set up skeleton class for Sprite
- Debugged player.java
- Updated UML diagram
- Javadocs

Kevin:

- Created skeletons for subclasses of sprite
- Set up drawingsurface.java
- Screenswitcher interface

Beta Checkpoint:

Aviral:

- Updated javadocs
- Worked on collision detection in sprite.java
- Fixed concurrent modification errors in Game.java and redid platform randomization so it does not give errors
- Implemented projectile motion for player.java
- Added “infinite” falling to game.java so that platforms randomize again after the player falls out of the screen
- Added randomized angular platforms and made them faster to load in Game.java
- Updated UML

Kevin:

- Added menu screen and game screen
- Implemented screen switching interface
- Generated menu screen beautification
- Created algorithm to make platform generation seem the most random
- Built player shooting mechanism
- Established the screen wrap-around for player movement
- Updated Readme

Final Submission

Aviral:

- Added moving and disappearing platforms to the game in Game.java
- Fixed platform colors
- Added adaptive enemy population control to regulate game difficulty progressively
- Fixed a myriad of bugs in game.java
- Updated javadocs and UML diagram
- Implemented enemy shooting in a way such that collision detection is accurate and precise
- Fixed an issue with “phasing” in platforms (this took forever to figure out what was doing it)
- Cleaned code in classes for ease of reading
- Fixed freezing during platform and enemy generation
- Added pairs as a data structure to categorize types of platforms to make collisions and removing and adding platforms much easier. Adapted code in Game.java to accommodate this.
- Updated readme
- Changed some graphical elements of project with sprite wrapping

Kevin:

- Updated readme and java docs
- Implemented help screen to effectively introduce the user to the operation of our program
- Visually enhanced the user interface of our project
    - Three-dimensional mouse parallax effect with both menu and instructions screen
    - Sourced high quality background images of nature scenes
- Implemented various images for the sprites to be accurately depicted by
- Established adaptive camera speed so that it follows the player
- Repaired the player projectile-enemy collision detection system
- Facilitated the player’s complete triple life system
- Built the digital apparatus to generate platforms and enemies
- Added player abilities such as double jump, shooting, and cooldowns
- Established visual indicators for player health, player ammo, player cooldowns, and score
- Constructed the powerups and coded the multiple means of obtaining them
- Added different enemy attack patterns and different attack frequency

External Libraries:

Processing java

Shapes library

Screen switching structure from GamePhysicsDemoAP

Background images from unsplash

- List the group members and describe how each member contributed to the completion of the final program. This could be classes written, art assets created, leadership/organizational skills exercises, or other tasks. Initially, this is *how you plan on splitting the work*.

Initial plan:
Both people need to make significant contributions to Game.java because this is the main class that contains all of the game mechanics. The other classes are not as important but both of us will create and code them as needed.

- Give credit to all outside resources used. This includes downloaded images or sounds, external java libraries, parent/tutor/student coding help, etc.]