Project 2: Scrolling Game COSC 102

Learning Outcomes

This project is an application of *object-oriented programming*–the programming paradigm you have spent last month learning and practicing.

The provided code reuses the game platform we used for the Tetris lab. For this project you will create your own scrolling game. We will help you first implement the logic behind a scrolling game. Then, similar to the "doodle" scrolling game (you may have clicked on), you will extend the basic version to design your own game. Doing so requires you to choose a theme and narrative, creating or incorporating nice visuals that complement your extended game play so that it is both challenging and fun.

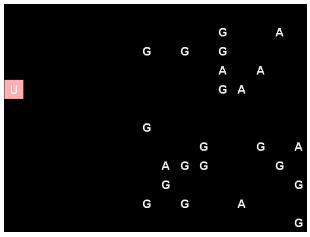
This game development project relies on an OOP organization provided by the game core code, which separates features among many classes. Your task is to effectively use the features available while integrating new omponents to build your own game.

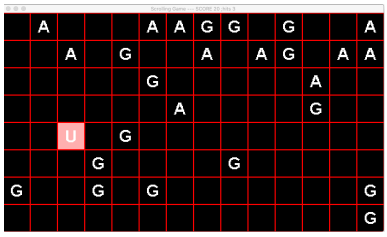
By doing so you are learning the power of OOP based on the following principles:

1. **abstraction**–generalizing an object, and its use in game development,
2. **cohesion**–being one thing or doing one thing well, which relies on grouping together code that contributes to a single task, and,
3. **encapsulation**–information hiding. The implementation (the internal workings) of each object type is separate, thus sufficiently hidden from the rest of the program responsibility.

To find out more about these principles and how they apply to game development read to the following **blog posts** written by Steven Lambert. http://blog.sklambert.com/introduction-to-oop-for-game-development/

1 Introduction





The above screenshot shows the provided demo version of the scrolling game in action. The game player controls the **U**, moving up, down, left and right while the other images move right to left towards them. Specifically,

Gs , standing for *GET*, represent assets to gather as each increases the game score; and

As , standing for *AVOID*, represent assets to keep away from, as collisions with them ultimately lead to losing the game.

The window displays a game board based on an underlying 2D grid. The placement and movement of all items, including the player, is tied to different "cells" of the game grid.

2 Overview and Provided Code

Your first task is to implement an un-themed version of the game, resembling the `ScrollingGameDemo.jar`.

2.1 Setup

The starting code and image assets for this project are contained in a `zip` file.

- Run the `ScrollingGameDemo.jar`, as you did with Tetris, and re-run it each time you are not sure about a requirement/feature.

  1. Press the `Enter` key, couple of times, to get to the game play.
  2. Press the arrows keys, `'D', 'Q' ...` to update the game.

  The only source file you should modify for Part I is `AScrollingGame.java`, but you should examine the other files to understand the code dependencies.

- Open the following three files

  1. `GameCore.java` is an `abstract class`.
  2. `AScrollingGame.java` is a child of `GameCore`. All the code you write for Part I should go in this file. Do an initial scan to compare its code to its parent.
  3. `GameLauncher.java` contains the `main` method. Examine its few lines before running the program. You will change the class constant `RUNNING` to launch your game version:
     - `RUNNING = BASE` to run a `AScrollingGame` instance (Part I: your replication of our demo)
     - `RUNNING = CREATIVE` to run an instance of your `CreativeGame` implementation (Part II).

2.2 Files Structure

More precisely, the five provided `.java` file have the following relationships allowing them to effectively implement the game mechanics.

1. `GameLauncher` contains the `main` method to launch a particular version of the game.
2. `GameCore` implements the game timing and rendering. More importantly, `GameCore` has an instance to control the underlying game board structure and represents only a partial implementation:
   - HAS-A `GameGrid` (composition/horizontal relationship) and provides methods for children classes to maintain and update the state of a game at particular `Location`s within game `grid`. Study and use its `Grid Methods`, as well as the class constants .
   - has many `abstract` methods required to be implemented by children to design a functional game.
3. `AScrollingGame extends GameCore` and will implement an un-themed scrolling game, replicating the demo. In effect,
   - `AScrollingGame` IS-A `GameCore` (inheritance/vertical relationship). You must rely on its parent methods to execute the main game loop and control the game board grid.
4. `GameGrid` actually updates and renders the content of the game 2D board. You should not have to directly call its methods, but use the appropriate `GameCore` methods to make changes in your `AScrollingGame` code.
5. `Location` is used to refer to an actual coordinates pair, (row, col) within the 2D `GameGrid`. Note that the upper-left corner of the window represents the coordinate $(0, 0)$.

   `null` signifies that no asset is present at a location.

3 Part I – The Basic Scrolling Game

Completing Part I requires you to implement the `AScrollingGame` code. Specifically, you should implement the following features, as illustrated in the `ScrollingGameDemo.jar`.

3.1 Features

- The player should be able to move up, down, right and left in the grid, according to arrow key presses.
- Each render tick
  1. a far-most edge should be **populated** at random with either types of assets (Gs and As) or nothing, and
  2. in a vertical or horizontal direction (your choice) **scrolling** must occur. For example, in the demo each column from the right scrolls one to left.
- Collision should be implemented such that the player always remains at their location. Collision might occur in at least two occasions
  1. when the player moves and
  2. when scrolling is taking place
- The scoring is updated according to collision. When collision occurs with
  - a `"G"` the game's score is increased by 10
  - an `"A"` the total hits is increased
- The game is over when either
  1. the score reaches 100, a 'win' or
  2. the total hits reach 10, a 'loss'.
- The following user interface affordances are available:
  - `'q'` quit the game,
  - `'p'` pauses or resumes the game play
  - `'s'` takes a screenshot of the game grid
  - `'9'` speeds up the game play
  - `'0'` slows down the game play
  - `'r'` reset the game speed to its default speed
  - `'d'` makes visible or invisible the game grid

Getting Started – As in lab, run the demo and observe its state. Based on your observations from its initial state, think about the simplest next step. Make sure to have a plan for the next smallest item so as to incrementally develop your code. Implement and test one feature at a time (always debug the smallest step you can conceive), then move on to the next.

3.2 Checks

Make sure of the following is taken into account in your implementation.

- A path through the grid exists without colliding with an "avoid" for the player to navigate through.
- Constants are defined and used as appropriate. Use and model on the ones provided.
- Instantiating objects is costly. So make sure to use the `Random` variable we provide for example.
- The size of the game is customizable at creation. Make sure your code works for different grid size values.

3.3 Milestone

As a milestone, you must

1. submit on Moodle all the `.java` code you updated, and images if applicable. You should at the minimum submit your `AScrollingGame.java` , and
2. share a less than 2 minutes video, Project2_milestone_name. Preferably recorded not in your room; it is not a demo but a reflection. Tell me about your experience learning from this Part I, feel free to share your pride and/or frustration, and tell me what you are planning for Part II. If you need help, please let me know.

Submit your code by **Friday, October 30, 2020, at 6:00PM**; and the video by the next 24 hours.

4 Part II - The Creative Game

Now it is up to you to design and implement your own game. Your game must have a theme. You can be as creative as you like, as long as your game creates a theme which expands upon the Part I `AScrollingGame`. You should focus on making the game interesting, fun and attractive. Be sure to choose a theme for your game that is *appropriate for a general audience*.

Note: Games that only replace graphical elements of the prior version are not sufficient. You can make pretty incredible games (you managed Tetris!) so I look forward learning from you all.

4.1 Setup

1. Make a new class called `CreativeGame`. This class should extend a previous class, it is up to you to choose between the abstract and the other version.
2. Change the constants in `GameLauncher.java` to run your creative game.

4.2 Requirements

You must design the following elements to complete your own creative game.

- The game's theme/story including its title, splash-screens, cell images, and anything else that determines the game's theme and playability.
- A scoring mechanism as well as the win and lose conditions. Games that keep running forever are not appropriate. The player needs to be able to win in a reasonable time frame.
- A splash-screen, instruction and end screens make the game nice, easy to understand and user-friendly.
- Some additional features or mechanics, i.e. rules and interactions, that expand on the design of Part I to make your creative game interesting. Specifically, implement at least two unique game mechanics: each new aspect to your game's design will add new rules or features that impacts the way the game is played. Some examples include new power-ups, new levels, different types of objects, different win/lose conditions, new abilities for the player, a progressive storyline, etc. For full credit, the new features you add to your game should be creative and interesting.
- Tune your game to make it fun to play. The speed and difficulty should make the game playable, which means both winnable and challenging.

5 Code Design

You are required to submit and adhere to the following:

- Provide a `readme`, i.e. plain text file, that outlines your theme, customizations, and overall vision for the game. Explain the design for the implementation of your extra features: the methods and algorithms modifying data state (variables, maybe structures or classes). Use it for your final video.
- Use constants (instead of magic numbers or hard-coded values), meaningful variable names and helper functions with a succinct header comment for each.
- Since this is a complex program, make sure to revise each part of your implementation. Your code style and design should make it easy for a reader to find and understand the different components of the implementation. A classmate should be able to trace each of the extra functionalities you implemented to make-up your unique game.
- Your code should be of quality design and continue to match the standard used so far in the course, in lectures and previous coding exercises. You should refer to the coding guidelines document posted on Moodle

6 Submission

Submit on Moodle the following

- Final source code for your game. Submit all the `.java` files.
- Image files required to implement your theme.
- Two screen captures you took with the `'s'` key to showcase your game features.
- A `readme` file.

Share a video of you playing your game and commenting on its implementation.

> Submit your code by **Friday, November 13, 2020, at 6:00PM**; and the video by the next 24 hours.

7 Grading

Your project will be graded as follows:

- 20% Checkpoint Submission (5% for video)
- 50% `AScrollingGame` Implementation
  - 40% Correctness
  - 10% Program Design
- 30% `CreativeGame` Implementation

Image Tools

The image type that works best with this game is a GIF file. You are allowed to find good images on the web as long as you document your source and URL in your `readme` file.

To make your own, below are some advanced software you may want to look into for making or editing images.

Gimp https://www.gimp.org/downloads/

Inkscape https://inkscape.org/en/download/

YouIDraw http://site.youidraw.com/, which seems simple and powerful

Sketchpad https://sketch.io/sketchpad/, which seems interesting but may not handle transparency directly

There are many free tools online that you may find simple to use, and there are several free icon and image galleries from which you can obtain images. Don't hesitate to talk to us and classmates if you have difficulties. I encourage you to recommend software to your peers and solve image software problems collaboratively.