Game Design Project

Now you get to build your own game! A skeleton structure is provided but the rules and content of the game are up to you. You may also choose to come up with your own type of game which is very different from the structure provided. Check with the instructor early to confirm that your game idea will meet all of the necessary criteria.

1 Background

Download the game from the course website: `https://projects.cs.dal.ca/hallab/CSCI1106` (This document was packaged along with those files.) In the game folder you downloaded, you should find the following:

`aidRunner.fla` :
: This is the main file for the game. Open this in Flash and choose the Quick Project option in the Project panel to create the Flash project for the game.

`Main.as` :
: This is the code to run the game, which is linked to `aidRunner.fla`. When the game runs, this code runs first (like the `Main.as` file in the Brick Breaker game).

`Level1.as` :
: This is the code to run the first level of the game. It is the class for the *Level1 MovieClip* object which you should find in the aidRunner.fla files library.

When you run the game, you should find that the first level begins by placing the player object at the left-center of the stage. The terrain scrolls continually to create the illusion of flying over land. Meanwhile, objects appear randomly at the top of the stage and fly leftward toward the player at varying velocities. At this point, this is all there is to the game. You determine the rest.

When you inspect the aidRunner.fla file, you will discover several objects in the library:

Terrain :
: This is essentially the green land formation that scrolls to the left in the first level. Since it is a *MovieClip* object, you can detect collisions with it.

GameObject :
: Instances of this object are randomly placed on the stage as the game progresses.

Level1 :
: This is the container for everything to do with the first level. The code for this is in the `Level1.as` file.

Level1Panel :
: This is the object on the upper-left corner of the stage in the first level. Currently, it has the words Level 1 at the top and is a translucent red block framed in black.

Player :
This is the player object. Right now, it doesn't do much.

In the `Main.as` file, you will find code to begin the game by loading the first level and an event listener that calls a function to check whether the game is over at each new frame. What happens when the game is over is not yet determined. You will place the code to direct this into the `if (gameOver)` block of the `checkGameState` function.

In the `Level1.as` file, code is included for placing the player on the stage, creating and moving the terrain, placing game objects on the stage, moving game objects, and moving the player. Coding the players movement is left up to you. You can add the code to the `movePlayer` function.

You can also modify how the game objects appear and move in `Level1.as` but some code for this is already given in the `addGameObjects` function. This function is called every time the screen refreshes. Game objects are not added at this rate though. This is because of the use of random numbers. `Math.random()` gives back a randomly selected decimal number between 0 and 1. We multiply this by 100 and round it up using `Math.ceil()` to get an integer in the range of 1 to 100. If that number happens to be less than or equal to 1 (**probNewObj**), a new object will be placed on the stage. In other words, there is a 1% chance that a new object will appear in the game each time the screen refreshes. Feel free to adjust this value or even to change it as the game progresses.

The x and y coordinates and velocity of each game object are also determined by random variables. The y location of the object can be anywhere above the terrain. The x location of the object is initially set just beyond the visible portion of the stage. As the object moves, it gradually becomes visible. Objects are originally placed out of sight of the player so that they do not appear suddenly on the stage. Finally, the objects velocity is a randomly selected number between 1 and 3. This is the number of pixels this object moves in the y direction in each frame.

The terrain is just one very long polygon created in the constructor of the *Level1* object. It adds a new elevation at every 100 pixels and does this 100 times leading to a polygon that is 10000 pixels long. This can be modified in the for-loop to change the nature of the terrain, possibly making it easier or more difficult to navigate.

2 Your Task

Now you need to design and implement the rest of the game. Some questions you must answer are:

- How will the player move? With the mouse? With the keyboard? Can the player move in all directions or are they limited?
- What is the objective of the game? Will you include enemies / negative game objects? What about rewards / positive game objects?
- How will the player win / complete the level? How will they lose?
- How will the player know how well they are doing as they play?
- How will additional levels differ from the first one?
- How will you communicate the purpose, rules of the game to the player? How will the player know how to play?

Of course, you are free to modify the appearance of the *Player* and *GameObject* objects and to add more if you like.

Be sure to document your work as you proceed. In addition to the completed game, you will need to submit a Technical Manual and User Manual at the end of the module. The details of these are provided in the report guidelines.

3 Grading Scheme

Use the following rubric to guide you as you develop your game. In other words, if you still have not added the code necessary to move your player, do not spend all class to make the game look cooler. Save the special graphics, sound effects and other features until you have a working game.

F	No changes made to the game.
D	Some modifications were made to the original project files but the game is still not playable.
C-	Player movement is successfully added to the game.
C	The game tracks collisions between the player and game objects and responds to them.
C+	The game has a clear objective (including the use of positive and/or negative game objects).
B-	The player is able to easily track their progress through the game (in the form of points or some other measure / approach).
B	Winning and losing the game are both possible and the game rules, purpose and how to play are clearly communicated to the player.
B+	The game includes multiple levels which increase in difficulty.
A-	The game has some polish (looks OK) and includes audio effects.
A	The game is polished (looks good) and has some interesting special effects.
A+	The game is highly polished (looks really good) and is compelling.

Each grade cumulatively builds on the criteria for the lower grades. Therefore, to achieve a grade of C, your game must not only respond to collisions between the player and game objects but also enable successful player movement. This represents half of your mark in the project. The other half of the grade comes from your technical and user manuals.

If your game design is non-standard in some way (for example, the player does not move), you must make a special case for this to your instructor early in your work on the game. The lack of a required aspect of the game may be permitted but only if the request is made before the final project lab period. Otherwise, you will be graded based on the rubric above. In other words, if you have a stunning game but the player does not move, you will only be given a D. Therefore, you must design at least the basics of your game and be able to answer the questions listed above before you begin to program and modify the project files.

4 Project Report Guidelines

At the end of this module, you will submit a Technical Manual (80%) and a User Manual (20%). Together, these will document the game that you developed for your project. The detailed requirements for each of these are given below.

4.1 User Manual

This is the written up instructions for how to play the game. If you were to sell your game, this would be the manual your users would read in order to know how to play and what to expect. The

user manual should comprise the following sections:

Title Page : This page should contain the name of your game, the names of the developers, and a screen shot from the game play. This is the cover of your manual. It should get the users excited about playing the game.

Game Overview : Use at least one paragraph to describe the purpose of the game. For example, in the Brick Breaker game, the purpose is to break the bricks by bouncing one or more balls off of the paddle. Describe how this purpose relates to winning the game. Show screen shots here.

Rules : Describe what must happen in the game and the consequences of this not happening. For example, the ball must bounce off the paddle and not off the bottom of the screen in the Brick Breaker game. The consequences of the ball hitting the bottom of the screen were the loss of points. Also use this section to describe how the player gets points (if you use points in your game).

How to Play : In this section, provide the player with specific instructions on how to operate the player and, if you have multiple levels, describe the differences between these. Describe the different sections of the screen and what they mean. For example, if you have enemies in your game, include images of them and describe how the player interacts with them.

The user manual must be *no more than 3 pages in length*, including images and the title page.

4.2 Technical Manual

The purpose of this document is to provide a detailed description of how your game works behind the scenes: the code. It should be clear enough that someone who has never worked with your game but whose ActionScript skills are at the same level as the average person in the class would be able to use this manual to continue to improve your game. Imagine you have just finished an internship with a game development company and your project will be passed on to another employee. Your technical manual must provide the details that employee would need to continue to develop your game into a success. The user manual should comprise the following sections:

Title and Author Information : The name of the game should be the title of this manual. Along with this, you must list the names of the game developers (you) and the current date.

Introduction : Provide a brief description of the game–the goals, how it works. Not many details are required because the User Manual will explain all of this in greater depth.

Description of `aidRunner.fla` : Use at least one paragraph to describe each of the objects found within this file (for example, *Player*, *Background*, *Level1*,). In other words, use one paragraph to describe the Player object and one paragraph to describe the *Background* object. Do not merge the descriptions of multiple objects in the same paragraph.

Description of `Main.as` : at least one paragraph to describe the class variables and constants (private and public) and use at least one additional paragraph to describe how each function works and how it connects to the rest of the game.

Description of `Level1.as` : Descriptions of the variables, constants and functions should be given in the same level of detail as was done for `Main.as`.

Additional Sections for any Additional Files : Describe the content of these files in the same manner as you did for `Main.as` and `Level1.as`.

Future Work : Describe what could be done to expand this game. If you have any errors that you could not fix, use at least one paragraph to explain each of these.

The technical manual must be *no more than 7 pages in length*, including images and the title page.

In both the user manual and the technical manual, standard conventions for grammar, word use, spelling, citations, headings, paragraphs, figures, and tables are expected. Templates of these manuals are provided on the course website (`https://projects.cs.dal.ca/hallab/CSCI1106`) so you know how they should be formatted.

Both manuals will be marked using the following rubric[1]:

	Exceptional: A	Acceptable: B	Substandard: C-D	Unacceptable: F
Content and Structure (50%)	Contains all required information. Ideas well organized and logically laid out always or almost always.	Contains most of the required information. Ideas well organized and logically laid out with competence.	Contains some of the required information. Minimal organization and logical progression of ideas.	Is missing most of the required information. Little or no organization or logical flow of ideas.
Analysis and Depth (30%)	Identifies and explains all issues and design decisions. Considers the issues from multiple points of view. Shows superior understanding of subject.	Identifies and explains most of the issues and design decisions. Shows commonplace understanding of subject.	Identifies and explains some of the issues and design decisions. Shows partial or limited understanding of the subject.	Identifies and explains few of the issues and design decisions. Shows a great deal of misunderstanding about the subject.
Presentation, Style & Tone (20%) Standard conventions for grammar, word use, spelling, citations, headings, paragraphs, figures, and tables. Word choices set appropriate style and tone.	Always uses standard conventions. The document looks professional. Shows exceptional use of tone and style. Speaks to the reader with precise, concise, appropriate language, and choice of words.	Mostly uses standard conventions. The document could use some editing. Shows competent use of tone and style. Makes good word choices.	Does not consistently use standard conventions. The document requires significant editing. Shows minimal attention to tone and style. Shows poor usage or ineffective word variation.	Standard conventions are flouted. Document is unreadable. Shows little or no understanding of appropriate tone. Uses inappropriate language and word choice.

5 Deliverables

There are three deliverables:

Game contained in the `.swf` file
Technical Manual in PDF or Word format
User Manual in PDF or Word format

The *game* is due at the beginning of the presentation period for this module because you will be presenting it to your peers during this period. The technical and user manuals are due at 8:30am on April 11, which is one week after presentation day. All files must be submitted to your TAs (please confirm email with them) and the reports must also be submitted in hard-copy.

[1]Based in part on Fleming, "Grading Rubric for Written Assignments", CSCI 2100, 2011