**Major Programming Project**

Here is your opportunity to search for a problem that you think can be solved using a structured programming in Java. You will go through all the steps of trying to find a problem to solve, and to step through all steps of solving a complete solution. Not only will you be solving a problem, you will be documenting all the specifications of the program as well as how these specifications change throughout the exercise.

Your project will include programming techniques that make use of the best form of file input/output required to meet your project specifications. You might ask: "What if my project does not need any form of file input/output?" **It will**.

**Object Programming:**

Much of this major programming project will focus around your ability to embrace object-oriented programming concepts that will be introduced throughout the term. That means that you will be starting this project, and then you will have to adjust your code as new programming concepts are introduced. This will upset you sometimes, but I can live with that. The clearer your code is, the more easily it will port to new structures that are introduced throughout the term.

**Asana:**

In this class, you will be using cloud-based project management software“Asana.com” to journalize your entries. You will be creating projects that will have their own **calendar**. You create “tasks” that get added to the calendar. The name of the task should clearly describe its mandate. The top of your task will have a link to your GitHub repository so that I can clone it at anytime. If your project does not work, still provide a link to your repository. Within the “task” you can journalize the comment that it does not work. Once the task works to your expectations, mark the task as “Complete” with a check mark.

Asana will be used in lieu of a journal, so you should be sharing your progress and frustrations on a daily basis – within the task that you are working on. All members of the programming team will contribute to the comments within each task. One member may have primary ownership of a task, but you are a team, and team members are assumed to support the team.

The multiple submission process will give us a chance to make sure your experience is as constructive as possible. My support notes are functional, but not great, that is why I consider this project to be as much a research project as a programming project. That leads me to the next section:

**Research and referencing your sources:**

Most programming is learning from other’s examples. I would expect most of you to be looking online for examples of how to approach your solutions. I also believe that copy/paste is sometimes more effective than re-typing what you see on the web. Just like in a history essay, you provide your teacher with sources of your quotations and numeric support, I will ask for the same. **When you embed any pieces of code from someone else, please comment the web site that helped you.** You will also have a list of sources on one page at the end of your project.

**Project organization:**

Your project will build with a new branch per release and a new branch for each scratch program. You will have a folder called “documents” within your project where all of your documents will be updated.

**Submission deadlines:**

Since each submission requires a meeting/debriefing with me, I will group the class into groups A, B, and C. This will allow me to give quicker feedback after you submit. When you look at the due dates, the first date is for group A, the second date will be group B, and the third, group C.

|  |  |  |
| --- | --- | --- |
| Weight | Description | Due Date |
| 8.5/10 | Initial Specifications and release schedule | Friday, Sept. 7 |
| 10 | Work in Progress Report 1 | Friday, Sept. 14, 21, 27 |
| 10 | Work in Progress Report 2 | Friday, Oct 5, 12, 19 |
| 10 | Work in Progress Report 3 | Friday, Oct 25, 2, 9 |
| 10 | Work in Progress Report 4 | Friday, Nov 15, 23, 30 |
| 10 | Work in Progress Report 5 | Friday, Dec 7, 14, 21 |
| 20 | Presentations: | Jan 14-18 |
| 50 | Program/ User Manual/ Final Package/ Journal | Thurs, Jan 24 |
| 130 | TOTAL |  |

You will be required to keep all your work in one folder. When it is handed in, I will be making sure your have followed through from your proposal all the way to your presentation.

**Initial Specifications**

You will address explicitly how the user will input information, the possible choices presented to the user, and the forms of output for the user.

You will also create a “**release schedule**”, which will list all the steps that you will go through on your way to a complete project. This schedule will focus on incremental features of to project, **not specific time lines**.

# Work in progress reports

This will be your progress report of how your program has developed. Here, you will summarize your major breakthroughs, major challenges, and modified plans for the future. This report will be printed out so that I can write “suggestions”. I expect you to follow those “suggestions”, or give me strong rationale as to why you disagree, or show me code as to why you could not make my suggestions work. In a world of lots of copied code, modifying expectations is a great check on plagiarism. I expect you to copy some code from others, and incorporate it into your work – with proper sourcing. I also expect you to make modifications in accordance to my “suggestions” as a further way of showing you understand and can modify other’s code. I will keep these printed reports with my written suggestions. You can take a picture of the report as a reminder of the directions I gave you.

**Scratch/test code:**

Within your WIP report, I expect you to have one isolated program that tests a concept that you want to integrate into your major program. The mandate for this program will be limited, but it should be created with the intent to test a concept out without screwing up your big program. This program will be a “branch” in your GitHub repository. Clearly name this branch - don’t just call it “Scratch1” - that does not describe the mandate.

**Peer Assessment**

At the bottom of each WIP, you will have a group discussion regarding the input of each team member. If a team member is being carried by the others, this is your chance to share this issue with the group. There will be a numeric weighting that I will use to assess the merits of each team member.

For example, if there were 3 team members, there will be an allocation of 300 “peer marks” to be allocated between the members. If one person is doing more work, and one is doing noticeably less, the marks may be allocated as follows:

Leader: 110

Worker bee: 100

Slacker: 90

This discussion will take place with EVERY WIP so that all members share their understanding of each other’s contribution. I will use this information when it comes to mark each individual.

**User Manual/Final Package**

I will be marking your **entire package**. Your entire package should be organised in such a way that I can easily check the development from your original specifications to your final product. This is to help you, as a student just in case your presentation does not go well. I will be able to see how much work you did in order to create the program and presentation. You will electronically hand in EVERYTHING, including all of your source code, journal, manual. When I try to run your program from the source code, and the compiler complains about missing files, I get very sad – and you lose marks.

INITIAL PROJECT SPECS

NAME:\_\_\_Matthew & Justin \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PROGRAM OVERVIEW:

The idea of our game is a side scrolling infinite runner. You will be playing as lonely pancake trying to make save planet of Ihop from getting taken over by Ihob.

DESCRIPTION OF USER INPUT:

The movement for the game will be WAD, W to jump, A to slow down, D to move the right. There will also be one power up, it will be chocolate powerup. This power up allows the pancake to throw chocolate chips at enemies.

DESCRIPTION OF PROGRAM OUTPUT:

There will be two types of enemies, first type will be projectile enemies: Fries, they be throwing objects at the protagonist. The second type is non-projectile: Burgers, the will be running at the protagonist. There will also be pit holes in the ground that the protagonist should be avoiding. If the hero lands on these holes will go to the start screen.

STRUCTURE OF YOUR DATA THAT WILL BE STORED IN TEXT/XML/JSON FILES

There will be a high score and a timer for how long you have lasted in the game.

**Release Schedule (the second half of Specs)**

**Here, you will show an itemized plan for completing your project. You will share how you foresee your program developing, from 1.0 to 6.7 (be creative with release names). Do not worry about specific times of completion; be more concerned about the specifications that would make a given release clearly unique. The release schedule will be like rungs on a ladder that will give you direction with clearly defined progress**ive goals.

|  |  |
| --- | --- |
| **Menu screen** | **Start with a starting screen with a play button.** |
| Game over screen | Here we will try to get the game over screen for after our character dies |
| Sprite | Sprite for characters such as the pancake, fries, and the burgers |
| **Background** | **This where the scrolling background will be made** |
| Making a Pancake | Here we will try our best and draw the pancake |
| Loading in the Pancake | Here we will load in the Pancake into the game |
| Movement for pancake | The movement for the pancake is going to be made here |
| Making the Burger | Here we will make try our best and draw a burger |
| Loading in the Burger | This is where we will just get the burger in our game. ( No movement) |
| Movement for Burger | The movement for the Burger is going to be made here |
| Making the Fries | We will draw some fries |
| Loading in the Friesi | This also is where we will just load in the fries |
| Movement for fries | The movement for the fries is going to be made here |
| Working on gravity | This will be a where we will make gravity for our game |
| Projectiles | Enemy projectiles such as the fries flying will be worked on |
| HitDCT for pancake | Here we will work on the hit detection for the pancake, so that when it hits  a pothole the screen will go to game over. |
| HitDCT for burger | Here we will work on the hit detection for the burger, so that when it hits the pancake it will print hit pancake |
| HitDCT for potHoles | Here we will be working on the hit detection for the potholes. If the Pancake hits the pot hole then the game will reset |
| Timer + score | Here we will have our timer and the score. |
| Power ups | The pancake will be able to shoot chocolate chips by collecting chocolate cookies |
| chocolate chip | This is where we will be working the power up that allows our character throw chocolate chips |
| Shop | The shop will have some new characters you can choose form, new kind of pancake. |
| Making shop button | Have new customs for the player to chose form. For example: Different kind of Pancake |
| Making a mode button | This where the speed of the game will change, or there will be more pot holes then in the normal mode |
| Working on making the game faster for the modes | This where we try and make the level speed up faster. |

**Work in Progress Report**

Major developments/breakthroughs(reference specific code please):

Some of our major breakthroughs so far have been making multiple screens for our game and and getting the scrolling for the background of the game working.

Major Challenges/setbacks( reference specific code please):

Our setbacks have not been coding aspect but the partner sharing aspect. Its mostly been on the GitHub and GitKraken. For example, when we would pull through GitKraken, sometimes a folder would get over written and we would lose very important code. So we would have to start over on the scratch that was overwritten. This has not allowed us to start implementing our scratches into one main project. That is why we don’t have one release yet. Also we would like to apologize for how our folder structure and that we will improve it before the next WIP is handed in.

Any modifications to your specifications/release schedule:

Yes there have been some modifications to our release schedule. We are a bit behind on it due to the problems stated above. Once we sort everything out we will catch up on completing everything.

**Description of your scratch/test program:**

Describe the generic concept you needed to test out:

The generic concept we tested out was making a menu, making the background of the game and making it move. Right now we’ve got a test menu that switches to the game screen when you push play, along with a background that moves vertically, and we are trying to make it move horizontally.

Source any web site/book that helped you with that concept:

Justin:

<https://github.com/JustSaiyan/Flash-Fire-FINAL-/blob/master/core/src/gdx/menu/Screens/ScrMenu.java>

<https://github.com/JustSaiyan/Flash-Fire-FINAL-/blob/master/core/src/gdx/menu/Screens/ScrPlay.java>

Matt:

<https://github.com/libgdx/libgdx/wiki/Spritebatch%2C-Textureregions%2C-and-Sprites>

<https://stackoverflow.com/questions/42824168/how-to-extend-the-background-in-libgdx-for-top-down-game/42842432#42842432>

Describe the code and the lesson that you learned from it:

Matt:

The whole goal is to get the two pictures of the background, and for them to rotate so it looks like the screen is scrolling. Here I get two images

Background1 = new Texture(Gdx.files.internal("floor.png"));

Background2 = new Texture(Gdx.files.internal("floor.png"));

From there I set the y max point to yMax = 1280; and coordinates for backgrounds one and two yCoordBg1 = yMax\*(-1); yCoordBg2 = 0;. From here I make the two images switch spots each time to make it look like it actually scrolling. yCoordBg1 += BACKGROUND\_MOVE\_SPEED \* Gdx.graphics.getDeltaTime(); yCoordBg2 = yCoordbg1 + yMax; // We move the background, not the camera

if (yCoordBg1 >= 0) { yCoordBg1 = yMax\*(-1); yCoordBg2 = 0; } batch.begin(); batch.draw(background1, 0, yCoordBg1); batch.draw(background2, 0, yCoordBg2); batch.end();

This all makes the picture switch spots so it look like it is rotating.

Justin: So far I have been trying to retrace my steps from last year and replicate my code for our new game (saves making new code, am I right?). Currently I have been able to implement the code for switching screens from our menu to our game screen

if (button == Input.Buttons.LEFT) {

if (isHit(screenX, screenY, btnPlay)) {

System.out.println("Hit Play");

gamMenu.updateState(1);

}

}

This code makes it so that when the “Play” button on the menu screen is hit with the left mouse button, it will update the game to be on ScrPlay which is designated to screen “1”, as seen

here: gamMenu.updateState(1);. I have also managed to implement music into our game

public ScrMenu(GamMenu \_gamMenu) { //Referencing the main class.

gamMenu = \_gamMenu;

Menumusic = Gdx.audio.newMusic(Gdx.files.internal("Menumusic.wav"));

Menumusic.setLooping(true);

Menumusic.play();

}

This code just makes it so when on ScrMenu it will play and infinitely loop the music. When implementing the code into our game the only thing I removed was the gamMenu = \_gamMenu;. This is because in our game at the moment this code is in our “MunchingTime” screen which is the main class and means we obviously don’t have to implement the main class as it’s already there. I just wanted to test this code out and so I will most likely add this code to seperate screens for when we have seperate music for the menus, main game screen, etc.

Technically I haven’t learned anything new but I am remembering and understanding how everything works with seperate screens amoung other things.

Describe any challenges that you enjoyed in integrating this scratch code into your major project:

Matt:

The most challenging thing must have been the brackets. I tend to put too many or too little, but it’s fun going back and looking for them. It taught me to take my time.

With each WIP, you will be submitting EVERYTHING. Organization is key. When I go to the group work folder**, I should see your project submitted in the following format:**

YourLastName: Under this folder will be the following folders:

**Asana Specs**: Your Asana calendar will have a task that contains a github link to your project and scratches. Please add any comments within this task that can give me a better understanding, like : “It does not work.”

Even if you provided the link to the same project in a previous task from a previous month – go big – add it again.

**Documents**: It will hold all of your documents: journal, WIP, Specs, Release schedule, list of sources, and all the other documents that will be submitted in your final project.

**Releases**: There will be a folder for each release, with one folder CLEARLY telling me that it is the latest, stable release.

**Scratch**: There will be a folder/ GitHub branch for each scratch concept that you tested before you integrated it into your final project.

**Peer Assessment: Matthew 100**

**Justin 100**

**Work in Progress Marking Scheme**

|  |  |
| --- | --- |
| Mark | Description |
| 10 | Great progress with release.  Clear scratch and integration. GitHub was clear and easy to navigate.  Solid documentation and list of sources – clear Asana journal. |
| 9 | Great work, but either scratch, release, or documentation is lacking. GitHub not well organized. |
| 8 | 2 of the 3 aspects are lacking. Poor modification of code to meet course expectations. |
| 7 | All 3 are lacking, or 1 is missing, or did not follow suggestions from previous report. |
| 6 | Very little progress shown, or did not follow suggestions from previous report. |
| 5 | Warning: this effort will NOT pass in final report |
| 4 | Little sign of work |
| 3 | Less sign of work |
| 2 | Project submitted with NO progress |
| 1 | Where is it??? I cannot give you a zero, even though you probably earned a zero. |
|  |  |

Major Programming Project Presentation: Agenda

1. Progression of your specifications. Brief overview of release schedule.
2. The solution: show all the great features of the program to the class: run your program!!!!
3. Disclaimer: known bugs for future reference.
4. Object overview: an overview of the classes you created and where they fit in your project.
5. Unique challenges and areas of code that you were proud of.
6. New programming techniques that were not taught in class. (Lesson)
7. Biggest frustrations.
8. Notes to future programmers of your project.

Major Programming Project Package: Table of Contents

1. Table of Contents: with a full listing of all file names(program, documents, presentation) and path. Make sure you clearly tell me which program to run.
2. Progression of specifications: initial specifications, release schedule, and Work in Progress reports.
3. List, and description of your test/scratch programs. There will be at least 5 of them.
4. Program overview/introduction.
5. Disclaimer/ list of known bugs
6. User Manual – your manual should START with clear directions of where your project is amongst all of the folders. Your manual should include screen captures to clarify your instructions.
7. List of sources that you used in order to make your program work.
8. Notes to future programmers of your project: directions that would make it better.
9. Lesson of your favourite programming technique that was not taught from my notes: consider this to be a more detailed description of your favourite scratch/test program.
10. Electronically submit **everything**, including your entire programming project.

Major Programming Project **Presentation** Marking Sheet

|  |  |
| --- | --- |
| Topic: | Presenters: |

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Description** | **Marks** | **Out of** |
| Presentation  Clarity | The presentation was clear, and easily understood. |  | 10 |
| Solution Demonstration | The class was easily able to follow how to work your program. |  | 10 |
| Bugs | You shared interesting known bugs, and demonstrated how to initiate them. At presentation time, it will be too late for you to handle insightful suggestions from the class as to how to fix these bugs. |  | 10 |
| Object Overview | Your audience was able to see how you structured your hierarchy of classes in a clear, logical manner. |  | 10 |
| Challenges/  Frustrations | You shared programming challenges that your peers will find humorous. |  | 10 |
| Lesson | You showed a part of your code that went beyond what was learned in class. Embed your code into your presentation – do not show the code within the IDE. |  | 10 |
| Pace | Your presentation moved at the proper pace. |  | 10 |
| Content | Your balance of content between **programming code**, and project management gave the class a solid understanding of your entire programming experience. |  | 20 |
| Notes to future programmers | These notes give great direction to future programmers as to where there is room for improvement, and possibly how to fix the problems (if only you had more time!). |  | 10 |

|  |  |
| --- | --- |
| **Criteria** | **Comments** |
| Presentation  Clarity |  |
| Solution  Demonstration |  |
| Bugs |  |
| Object Overview |  |
| Challenges/  Frustrations |  |
| Lesson |  |
| Pace |  |
| Content |  |
| Notes to future programmers |  |

Major Programming Project Marking Sheet

|  |  |
| --- | --- |
| Topic: | Programmer: |

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Description** | **Marks** | **Out of** |
| Package  Clarity | The package was complete and in order.  All sections were clear, and easily understood. |  | 10 |
| Bugs, Specs and journal | All three were clear and insightful. |  | 10 |
| Program:  Challenging | The problem that you chose to solve proved to be challenging. |  | 15 |
| Program:  Clear front end | Your program interface proved to be clear to the end user: both for data input and program output/display. The **manual** was an excellent support. |  | 15 |
| Program:  Clear code | Your code was clear, concise, well-documented, and structured.  Clear **comments** throughout: especially to give an overview of a method, class, or frequently used variables. |  | 15 |
| Program:  techniques | You used the best programming techniques to solve the problem. |  | 15 |
| Lesson | You shared a new programming technique that was not in my notes. You clearly showed how it could be used, and why all high school programmers should be using it. |  | 10 |
| List of sources | Clear links as to where you learned and borrowed programming techniques. This will be an entire list of the sites that you used in order to make your program better. Clearly reference the parts of your code that benefited from these sites. Your code will have comments of your sources as well. |  | 5 |
| Notes to future programmers | These notes give great direction to future programmers as to where there is room for improvement, and possibly how to fix the problems (if only you had more time!). |  | 5 |

|  |  |
| --- | --- |
| **Criteria** | **Comments** |
| Package  Clarity |  |
| Bugs, Specs and journal |  |
| Program:  Challenging |  |
| Program:  Clear front end |  |
| Program:  Clear code |  |
| Program:  Techniques |  |
| Lesson |  |
| List of sources |  |
| Notes to future programmers |  |