**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.

**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. **Journalize** how upset you are. It makes for entertaining reading.

The multiple submission process will give us a chance to make sure your experience is as constructive as possible. I am not a strong GUI programmer. I like to focus on cs contest problems rather than GUI programming – but I know that is what my students enjoy doing. 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.

|  |  |  |
| --- | --- | --- |
| Weight | Description | Due Date |
| 10 | Initial Specifications and release schedule | Friday, December 9 |
| 15 | Work in Progress Report 1 | Monday, Dec 19 |
| 15 | Work in Progress Report 2 | Friday, January 13 |
| 60 | Program/ User Manual/ Final Package/ Journal | Thurs, January 26 |
| 110 | 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.

**Project Journal**

You will also be keeping a programming journal of the project. In this journal you will be entering your development of the program specifications, as well as frustrations with the process. You will be sharing any new programming techniques that you acquired while solving problems within your project. Some students try to “retrofit” their journal, spending the last day filling in a bunch of nonsense for me. Don’t. Anytime we have a “discussion” about the program specs, or when your program crashes for x days in a row – journalize it. When you reach a new level of your program that you are exceptionally proud of – journalize it.

# 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. Within one folder, you will be **submitting your entire program folder** with your summary report, so that I can run your program and check your code. **You will also be submitting your journal as well as your manual.**

**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. You might say “That’s not how I do things.” I will correct you on that.

**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. I will take these numbers that you provide under advisement when it comes to assessing individual effort.

**Expectations:**

With your specifications sheet as well as your Work in Progress reports, I will make “suggestions” as to where you should go from there. 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.

**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:\_\_\_Maddie, Nicole\_\_\_

PROGRAM OVERVIEW:

When you run the program you can select instructions or play. When you start the game you choose a character. Once you choose a character you enter a maze. You have to get to the door to win before the timer runs out. Through the maze there are potions. One subtracts 10 seconds. One puts you back at the start. One gives you 10 seconds. You can collect coins to increase your score.

DESCRIPTION OF USER INPUT:

-Arrow keys to move left, right and jump

-Mouse to choose character and press play

DESCRIPTION OF PROGRAM OUTPUT:

There will be a maze from bird’s eye view. You will have to navigate the maze, while collecting coins. Potions will also be in the way and may help or hinder. You will have to reach the door before the time runs out. If you win, the winning screen will display your score.

**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 progressive goals.**

|  |  |
| --- | --- |
| **Release Name** | **New incremental features of this release** |
| **1.0** | **Menu with play and instruction menu.** |
| **1.4** | **Play button goes to maze screen with character on it.** |
| **1.8** | **Instruction button goes to instructions.** |
| **2.3** | **Character moves with sprite sheet.** |
| **3.0** | **When character moves camera moves.** |
| **3.8** | **Hit detection on maze walls** |
| **4.0** | **Add door.** |
| **4.2** | **Door takes you to winning screen.** |
| **4.8** | **Add timer.** |
| **5.0** | **If timer runs out go to losing screen.** |
| **5.2** | **Add potions to maze** |
| **5.4** | **Make one potion subtract 10 seconds** |
| **5.8** | **Make one potion add 10 seconds** |
| **6.0** | **Make one potion put you back to the start** |
| **6.2** | **Add coins to maze.** |
| **6.8** | **Make coins count your score, and have score displayed on winning screen.** |
| **7.8** | **When you press play from menu screen, it allows user to choose character, and when you press play from that screen it plays** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Work in Progress Report**

Major developments/breakthroughs(reference specific code please):

Major Challenges/setbacks( reference specific code please):

Any modifications to your specifications/release schedule:

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

Describe the generic concept you needed to test out:

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

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

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

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

YourLastName: Under this folder will be the following folders:

**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 for each scratch concept that you tested before you integrated it into your final project.

**Peer Assessment:**

**Work in Progress Marking Scheme**

|  |  |
| --- | --- |
| Mark | Description |
| 10 | Great progress with release.  Clear scratch and integration.  Solid documentation and list of sources – clear journal. |
| 9 | Great work, but either scratch, release, or documentation is lacking. |
| 8 | 2 of the 3 aspects are lacking |
| 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 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. Journal for the project.
8. List of sources that you used in order to make your program work.
9. Notes to future programmers of your project: directions that would make it better.
10. 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.
11. Electronically submit **everything**, including your entire programming project.

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 |  |