Designer	Regan Music
Date	6/7/2020
Course	CS205 – Integrated Studio 3
Version	1.1

Bachelor of Creative Software

CS205
Integrated Studio 3
(NZQF Level 6, 30 credits)

Labs

35% of the total Marks

Project Start	**Week 1, Thu 23/07/2020, 12:30pm**
Submissions:	
Console Application	**Week 3, Fri 07/08/2020, 11:59pm**
Lighting	**Week 4, Fri 14/08/2020, 11:59pm**
Animation	**Week 5, Fri 21/08/2020, 11:59pm**
UI	**Week 7, Fri 03/09/2020, 11:59pm**
Additional Features	**Week 8, Fri 11/09/2020, 11:59pm**

Objectives

Integrated Studio 3 aims to prepare syou with knowledge & experience to help you create your own games or otherwise mobile or web applications. As you experience development and design in a range of different situations you'll come to understand C#, OOP, Unity Engine, and various other tools and best-practices. This experience will see you able to design and develop confidently on your own as well as in teams.

These labs are designed to ensure that you're familiar with particular practices that are necessary to practice first-hand as opposed to certain concepts which are learned easily enough in-class.

Tasks

This brief outlines several submissions that have their own due dates, each is a separate task and tests you on specific topics learned in-class. This brief is different from other briefs in that it will repeatedly see you graded on **Task Completion**, **Code Practices** & **Project Management**, any extra criteria will be outlined per-task. The following outlines generally what each of these aspects are asking.

Task Completion

This is a check that you've completed the task as per the description. It is not concerned with code-quality or project management.

Code Practices

This checks the quality of your submitted code. You're expected to show clean, consistently formatted code that's **named well**, **commented** or **self-documenting** (self-explanatory names), is **DRY** (Don't Repeat Yourself) and makes good use of C# **classes**, **variables** and **methods** while making appropriate use of not only primitive types (**int**, **float**, **string**, **bool**) but also built-in Unity types, variables and methods. Feedback will be given with the intention to improve your coding quality.

Project Management

You'll be shown how to organize & manage your project folders, files & assets in-class. This criterion checks that everything is named-correctly, organized into folders nicely and that folders/files normally ignored by a git-ignore file are not submitted (.vs, Library, Logs, obj, Temp, .csproj, .sln, etc.).

Version Control

Version Control via **git** is taught in-class. Unity projects should not be submitted directly to Moodle. Host them yourself in a **git repository** with your own account. Make the project **public** and simply **provide the URL to your git repository in a text file**. You don't need to submit additional files (like images) this way, but zip them up alongside this text file.

Please see the "Marking Schedule" section later in this document for any task-specific details.

Please see the "Submissions/Deliverables of your Work" section later in this document for details on what should be included for each submission.

Submission/Deliverables of your Work

Submissions are accepted only via **Moodle** and should be a **zip file** with the following content per submission:

Console Application

- A folder containing the ***.exe** and any required files for it to run.
- A ***.txt** file that contains a URL to a **public** git project where the **Visual Studio** project is stored.

Lighting

- A folder containing screenshots (***.png**, ***.jpg**, etc.) of your game-view, demonstrating day/night lighting variants for and interior and exterior environment.
- A *.txt file that contains a URL to a **public** git project where the **Unity** project is stored. Remember your **.gitignore** file!

Animation

- A folder containing the Unity build (***.exe** and any required files). Your animations will be tested by playing the build.
- A *.txt file that contains a URL to a **public** git project where the **Unity** project is stored. Remember your **.gitignore** file!

UI

- A folder containing the Unity build (***.exe** and any required files). Your UI will be tested by playing the build.
- A *.txt file that contains a URL to a **public** git project where the **Unity** project is stored. Remember your **.gitignore** file!

Additional Features

- A folder containing the Unity build (***.exe** and any required files). Your UI will be tested by playing the build.
- A report that outlines which 3 features you added. Support your explanations with screenshots.
- A *.txt file that contains a URL to a **public** git project where the **Unity** project is stored. Remember your **.gitignore** file!

Learning Outcomes

On successful completion of this course, students are able to:

1. Articulate an individualised learning journey through reflection on specialist contribution to a team software development project.
2. Produce a useable, fit-for-purpose software product to the required technical standard.
3. Identify and resolve issues and problems in an open and collaborative way.
4. Engage in innovation and creativity as software practitioners.
5. Integrate effective collaborative practices in a team-based software development environment.

Marking Schedule

Console Application (5 marks)

Criteria	Marks
Task Completion • **BullsAndCows.exe** launches and asks the player to choose a 3-5 digit round. • Feedback is given In response to player input. **Bulls** indicate a correct number in the right position, **Cows** indicate a correct number in the wrong position. • Feedback informs the player when they have completed the puzzle, this happens before the app closes.	**3**
Code Practices Code is well-formatted, DRY, follows a consistent standard, well-commented or self-documenting, and makes the best use of OOP practices where possible.	**1**
Version Control Project files are stored in a git repository. History shows that the repository has been active since the project began and has been frequently committed/pushed to throughout the project (at least daily).	**1**
	5

Lighting (7 marks)

Criteria	Marks

Task Completion
• Screenshots are provided showing 4 scenes lit in Unity. There's, both, a day and night variation for an interior and an exterior scene. Images will be checked against the Unity project.
• The Unity project shows 4 scenes that match the provided screenshots and fit the following criteria:
  ○ Static objects marked as static;
  ○ Light Probe Groups & Reflection Probes provide global-illumination to dynamic objects;
  ○ Post Processing is used to enhance the scene.
	4

Aesthetic Quality The submitted screenshots look highly appealing and make good use of lighting, materials and post-processing to create mood and establish the correct time of day. The camera has been well-placed using the rule-of thirds or golden ratio.	**1**
Project Management Everything is named-correctly, organized into folders nicely and folders/files normally ignored by a **gitignore** file are not submitted (**.vs**, **Library**, **Logs**, **obj**, **Temp**, **.csproj**, **.sln**, etc.).	**1**
Version Control Project files are stored in a git repository. History shows that the repository has been active since the project began and has been frequently committed/pushed to throughout the project (at least daily).	**1**
	7

Animation (7 marks)

Criteria	Marks
Task Completion The **built executable** (***.exe**) runs and shows a character animating correctly when controlled via WASD controls. The following animations should occur correctly: • Idle; • Walk; • Run; • Jump; • Attack; **Marks are given by percentage of work completed.**	**4**
Code Practices Code is well-formatted, DRY, follows a consistent standard, well-commented or self-documenting, and makes the best use of OOP practices where possible.	**1**
Project Management Everything is named-correctly, organized into folders nicely and folders/files normally ignored by a **gitignore** file are not submitted (**.vs**, **Library**, **Logs**, **obj**, **Temp**, **.csproj**, **.sln**, etc.).	**1**
Version Control Project files are stored in a git repository. History shows that the repository has been active since the project began and has been frequently committed/pushed to throughout the project (at least daily).	**1**
	7

UI (8 marks)

Criteria	Marks

Task Completion
The **built executable** (***.exe**) runs and shows a scene with working UI. There is a HUD and a Pause Menu. See the following for details:
 	
HUD
• A player profile image with a radial gauge surrounding it for EXP.
• 2 Horizontal gauges:
  ○ Health;
  ○ Stamina.
Pause Menu
• 3 text buttons:
  ○ Resume;
  ○ Options;
  ○ Quit;
• Options switches UI panels and should have:
  ○ A volume slider;
  ○ A checkbox that enables super-fast mode;
  ○ A back button to return to the Pause Menu.
 	
UI will be tested in the built executable at the time of marking. A Unity scene will be provided, giving you everything you need to attach your UI to.
 	
Marks are given by percentage of work completed.
	4

Aesthetic Quality UI is formatted nicely in terms of space, size & colour.	**1**
Code Practices Code is well-formatted, DRY, follows a consistent standard, well-commented or self-documenting, and makes the best use of OOP practices where possible.	**1**
Project Management Everything is named-correctly, organized into folders nicely and folders/files normally ignored by a **gitignore** file are not submitted (**.vs**, **Library**, **Logs**, **obj**, **Temp**, **.csproj**, **.sln**, etc.).	**1**
Version Control Project files are stored in a git repository. History shows that the repository has been active since the project began and has been frequently committed/pushed to throughout the project (at least daily).	**1**
	8

Additional Features (8 marks)

Criteria	Marks
Task Completion The **built executable** (***.exe**) runs and demonstrates 3 features that weren't found in the template project. **As the code-base is large, you're be required to document your additional code in a written report with screenshots where possible. Undocumented work will not expect to be marked.** **Marks are given by percentage of work completed.**	**4**
Code Practices Code is well-formatted, DRY, follows a consistent standard, well-commented or self-documenting, and makes the best use of OOP practices where possible.	**2**
Project Management Everything is named-correctly, organized into folders nicely and folders/files normally ignored by a **gitignore** file are not submitted (**.vs**, **Library**, **Logs**, **obj**, **Temp**, **.csproj**, **.sln**, etc.).	**1**
Version Control Project files are stored in a git repository. History shows that the repository has been active since the project began and has been frequently committed/pushed to throughout the project (at least daily).	**1**
	8

Assessment's Policies

- Submissions are only accepted via Moodle.
- All work shall be properly referenced in the APA format;
- Refer to the Yoobee Colleges' current extension policy surrounding late submissions and assessment extensions. You will be provided with this via Moodle, see your tutors.
- Marks and feedback will be returned within three weeks of the submission date.
- Academic dishonesty and plagiarism are considered serious offences at Yoobee Colleges and significant penalties can be incurred, such as a reduction in the grade awarded for the assessment, failure of the course, and in some cases, suspension or expulsion from the College. Please refer to the **Student Handbook** regarding assessment **submission and plagiarism policy** for detailed information. By completing and submitting this assessment you are authenticating that the work is original and does not violate plagiarism or copyright law.

Marking Guidelines

Criteria	Individual Mark	Group Mark
Labs	35	N/A
Project Proposal	2	13
Unity Application	25	25
Total	**62**	**38**