

Lab Deliverables

Contents

Objective	1
Feedback and Final Grading	2
Recommendations and Preparations	2
Software Engineering Concepts Of Most Use	3
Design Pillars	3
Loops and Interactions	3
Primary Gameplay Loop	3
Secondary Gameplay Loop	3
Tertiary Gameplay Loop	3
Object Interactions	3
Example	4
Rubric / Grading Breakdown	4
Submission	4
Unity Research Experiments	4
Goal	4
Trello Cards	5
Ignore Presentation, Focus On Learning	5
Rubric / Grading Breakdown	5
Submission	5
Class Diagram	6
Goal	6
StarUML	6
UML Syntax	6
Parts Overview	6
Relationships	7
Rubric / Grading Breakdown	7
Submission	7

Objective

The objective of these lab deliverables is to help prepare you for and help you understand what it's like making a game with other people.

Feedback and Final Grading

For many reading this document this may be your first team project. To help you understand this process when a deliverable is due feedback will be given and time provided to correct the deliverable based on that feedback. Here is the general schedule for deliverables, which may be different based on holidays, breaks or other circumstances:

Day 2 – Start work on Loops and Interactions deliverable

Day 3 – Feedback for Loops and Interactions. Start work on Unity Research project.

Day 5 – Final grade for Loops and Interactions. Feedback for Unity Research project. Start work on Class Diagram.

Day 7 – Final grade for Unity Research project. Feedback for Class Diagram.

Day 8 – Final grade for Class Diagram

You will want to aim for completing a deliverable when feedback is given however final grades won't be graded until after feedback to allow corrections and more opportunity to learn from this process.

Recommendations and Preparations

Here is a list of things that will help you prepare for the tasks ahead of you.

- **Write down everything** – Keep notepad / Evernote / Google Docs open and note everything. You will never hold in your internal memory all the things that need to be done or kept track of. Learn to use external memory
- **Sort your time out** – Use a calendar, Siri, post it notes, whatever. Balance the tasks and your life
- **Observe your environment** – Take note of those that can get stuff done. How do they do it? As long as they don't go about by cheating, it may be worth stealing how they do the things they do.
- **People with experience, you are not the dictator of the group** – You have wisdom to impart; which is hard to do when everyone thinks you are a dick.
- **All the stuff learned in Software Engineering isn't just for a grade** – Pay attention, take notes on it, be ready to explain why and when it is good to use it and where. Especially for your game.
- **Things will change** – There is potential that every Course Director involved in the midterm will influence or even outright change parts of the project. That's alright.
- **Remember the scope of the project** – You will not be spending the next 3 months creating your own unique epic interactive experience. The focus of this midterm is not to reinvent the wheel.

Software Engineering Concepts of Most Use

Day 1

2D basics - Game Objects, Screen and world space, Collision, Version control, Scripting, Layers, Camera, UI canvas, Scenes.

Day 2

2D animation - Animation, Events, Finite State Machine (FSM), Transitions, Sprites, Prefabs, Camera Animation, FSM sub states, Audio, Caching components and object pools

Day 3

Messages and Events - SendMessage(), Delegates, Design Patterns (Focus on Mediator and Observer), Parallelism - Invoke & Coroutines

Design Pillars

Your first plan/architecture for your game will not be optimal but that's okay. Just the act of planning/architecting is critical for the success of your team's game. This will take time and discipline to accomplish. Some that have experience with this may feel that it is unnecessary for you, but this isn't about you. This is about the team. Take some time and work with your team to come up with 3 to 5 elements/emotions that the game is trying to explore and make the player feel. These design pillars will be the main focus for all the design decisions you make for the game. Write these pillars down in a place that's easily referenced, you will need them.

Loops and Interactions

With your design pillars in mind, think about all the events involved in your game. The things / activities the user does, the things the computer makes happen either in response or in spite of the player's activities. Separate these ideas based on time.

Primary Gameplay Loop

What do you want the player or other entities of the game to achieve on a second to second basis? Focus most efforts here. Bottom up approach.

Secondary Gameplay Loop

Minute to minute. Systems outside of and that feed into the primary loop would be here. Looking for depth as opposed to breadth.

Tertiary Gameplay Loop

Hour to hour. The why we keep coming back. Think about these only after you spent time on the others.

Object Interactions

Think about how all the objects in the game interact with each other and the resolutions of those interactions. Compile these interactions and their results into a grid. Be concise in your descriptions but detailed enough to understand what is happening. Refrain from using game specific lingo.

Example

<https://docs.google.com/spreadsheets/d/1dR7F6utKXpBalzIoeM00qr7doMG74hIH8ZFY612p6jc/edit?usp=sharing>

Have someone on the team make a copy of this spreadsheet and share it with the team so you can all edit it together. Post the share link to the #deliverables text channel on the team's discord as well so it can be viewed by your instructors.

Rubric / Grading Breakdown

Loops - up to 50 points

- 5 points – At least 12 secondary loops
- 10 points – At least 20 primary loops
- 15 points - Loop is in the correct section
- 20 points - Every loop has an understandable description

Interactions - up to 50 points

- 10 points - All game interactive objects are catalogued
- 20 points - Description of audio/visual feedback before/during/after interaction
- 20 points - Game logic is detailed for and from the point of view of the specific interacting object

Submission

The only thing needed to submit this deliverable is to provide the share link to the spreadsheet in the discord's #deliverables channel.

Unity Research Experiments

Goal

A common complaint of students in this class is that they do not have enough time to learn the tools being used in the project. In order to mitigate this, as a team use Unity to build a series of experiments focused around the project's mechanics, interactions, and all the things that happen on screen or under the hood. This will provide the advantages of learning how to use Unity, learn the effort needed to complete features, and allow you to grok the project.

Each team member must research a loop, interaction, or risk of the project (reference your previous deliverable for options if needed). The chosen topic of the research **must** include one of the following: movement, collision, a timer, input, or feedback (audio/visual). Researching more than one interaction is highly encouraged but not required.

Trello Cards

Once you have chosen your topics of research create a Trello card for each experiment on the 0c Architecture & Research board. Here is a link to an example card:

<https://trello.com/b/jEccrRsW/research-card-example>

Make sure you assign a team member to each card and that the card gets filled with at least three unit tests that will be used to confirm the implementation is successful. You will not be graded on whether you succeed in passing your tests or not but on the content of the card, the unity project and what you have learned. If you have failed to pass a unit test simply make a comment on the card about what you learned from the process and what you will do in the future to mitigate the risk of failure in the project.

Ignore Presentation, Focus on Learning

When creating your experiments, it is very important that you don't waste time with things not relevant to the research. **Do not** spend time looking for assets to make this experiment look better. Only use the minimum assets needed (You can get quite far with just colored cubes). This project is not about implementing a feature into your game but researching *how* to implement a feature. You are not going to use the content of this experiment in your game so do not spend any time on its presentation. Focus on the functionality or you will be wasting time. If you finish researching something early use the extra time you gained to research more topics! These experiments are all about quantity and not quality; leave the quality for the actual game.

You are welcome to reference tutorials online when researching but you must not use tutorial scripts/prefabs/scenes in your experiments. Your experiments must be created from scratch.

Rubric / Grading Breakdown

Trello - up to 50 points

- Detailed description of what all the experiment encompasses
- Checklist of multiple test cases proving the experiment is complete
- Comment covering what was learned and risk mitigation for failed unit tests

Experiment - up to 50 points

- Interactive experiment in Unity demonstrates all tests case attempts

Minimum one experiment per team member; Trello Card must exist for each experiment

Submission

Submission for this deliverable is at least one card per team member on the teams 0c Architecture & Research Trello board and a unity project per team member containing all experiments of that team member. These unity projects will be on the team's repository in folders identifying each team member.

Class Diagram

Goal

The purpose of this class diagram is to provide an overall view of all the scripts the team will need to finish the project, how they relate to each other and the dependencies on the Unity API.

You have not written these scripts yet so the focus is essentially only on those scripts you know for sure you will have in the game. It is ideal to treat this diagram as a living document that you can add and expand upon later if needed.

StarUML

You will need to use StarUML to create this diagram (download StarUML here: <http://staruml.io/>). This diagram is meant to be a team effort but because StarUML files can't be merged by git it is recommended to split the work. How you split the work will be left up to the team but often teams will have each member work on a portion of the game in a separate diagram and then combine all the members' diagrams into one before feedback is due.

A short discussion about how to use StarUML and an overview of UML syntax can be found here:

https://fullsailedu-my.sharepoint.com/:v:/g/personal/dbliss_fullsail_com/EbFKaTPLRVIHmAMgkLLl80MBdTrcZEKR GskDFgROUvtaFA?e=31s8I3

UML Syntax

UML follows a specific syntax for displaying the different parts of a class diagram. In addition to the above video you can familiarize yourself with the syntax using the following website as a reference:

<https://www.uml-diagrams.org/class-reference.html>

Parts Overview

Classes

Classes are all the scripts created for the game as well as the Unity components and classes that contain the Unity API methods you will be using in those scripts.

Attributes and Operations

These are all the member variables in the scripts created for the game. Do not show member variables of Unity components or Unity API classes because the focus is not on those classes but the classes the team will create.

Packages


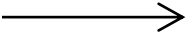
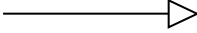
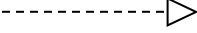
An object used to denote a group of classes that work together, often towards a key part or functionality of the game.

Note

An object used to give more detailed information about anything in the diagram. Can be linked to specific classes or other objects.

Relationships

Relationships are arrows on the diagram used to represent dependencies between classes and the strength of those dependencies. Below are the most common relationships found in class diagrams:

Dependency		when one class calls the function of the other via local variable (arrow points away from class calling the other)
Association		when one class has a member variable of the other class's type (arrow points away from class with member)
Generalization		when one class derives from another class (Arrow points towards the parent class. All unity scripts by default derive from MonoBehavior so do not show that relationship on your diagrams in order to keep it clean.)
Realization		when one class derives from an interface (arrow points towards the interface)

There are more relationships commonly used in class diagrams but because of the nature of C# using references for all user-defined objects and the garbage collector handling the memory of those objects, you will not need to use the other relationships on your diagram.

Rubric / Grading Breakdown

Classes - up to 20 points

- All scripts and referenced Unity API components represented

Behaviors - up to 20 points

- All variables and methods present in script classes

Packages - up to 20 points

- Packages are used to group all scripts with a common purpose.

Relationships - up to 40 points

- All relationships are represented and with correct relationship type.

Submission

The team will submit a **single** class diagram by committing and pushing a StarUML file to the team's repository.