

Project Outline

This is the basic outline for the GAME-EN semester project.

Please read carefully and make sure you understand before starting work on your project!

If you have any questions, feel free to ask [your lecturer or your INNO-LAB Summiteer team](#).

Contents

Introduction to the project.....	2
Technical Limitations.....	3
General.....	4
Supported libraries.....	4
Project Settings.....	4
Basic Folder Structure.....	4
Scenes.....	5
Scripts.....	5
Assets.....	6
Game Design.....	8
General.....	8
Players.....	8
How to set up your project.....	9
Checklist.....	13
Contact information.....	14

Introduction to the project

This semester you have the opportunity to participate in an interdisciplinary project with hundreds of other students and create a game together.

The aim of this project is to enable students from all game development skill levels to contribute to a single, working game without having to spend thousands of hours making one from scratch. To make this possible, a mario-party style game was chosen as the basic project premise.

Mario Party is a light hearted multiplayer family game. It essentially consists of two easily splittable parts, one being minigames - which aspiring game developers like you create - and the other being a board game that ties all the minigames together, which is currently getting developed by INNO-LAB teams.

The GAME-EN project for this semester is to create one such minigame, and add to a growing library of amazing games created by students like you.

Before you let your creativity run wild, make sure to read this document carefully, as it outlines a few things you need to consider when making your game, both from a [game-design perspective](#) and a [technical one](#). It's like a more advanced game-jam, where the design needs to be consistent and underlying technical elements need to be compatible with other minigames.

We are constantly working on improving the development experience for you. INNO-LAB teams are hard at work developing a framework that should solve most of the difficult challenges that minigame developers currently face. Until that is ready, you will have to solve challenges like player-join behavior or state-management yourself, but don't worry, if you get stuck, simply [contact your lecturer or the INNO-LAB team](#) and we will help you out!

With all that said, we hope you are just as excited as we are, and we can't wait to see what creative games you will come up with!

Technical Limitations

To ensure compatibility with the main project, certain technical limitations need to be followed:

Unity Version	2021.3.XX
Render Pipeline	Default Pipeline
Required libraries	Input System, V1.44 Installed by default: TextMeshPro, V3.0.6 Engineering Feature
Additional supported libraries *	Cinemachine, V2.8.9
Project Settings *	Player => Other Settings => Active Input Handling = Both
Basic Folder Structure	<ul style="list-style-type: none"> ● Imports ● YourMinigameName <ul style="list-style-type: none"> ○ Art <ul style="list-style-type: none"> ■ Audio ■ Materials ■ Prefabs ■ Textures ○ Code <ul style="list-style-type: none"> ■ Scripts ■ Shaders ○ Scenes
Mandatory scenes	<ul style="list-style-type: none"> ● MainMenu ● Tutorial ● MainGame ● GameOver
Assets *	FH Assets only
Audio	Use your own Audio (Open-Source)
Input	New Input System ("Input System" library)
Input Devices	Primary: Gamepad If possible: Mouse & Keyboard
Build Platform	Windows

* : Can be customized after consulting with the main project team

General

It is very important that the outlined technical limitations are followed. If not, your minigame might work in your specific project with your specific environment, but not outside of it. If two minigames use conflicting project settings or unity versions for example, it becomes impossibly difficult to import both of them into the same project.

Supported libraries

In addition to the Input System library, we also support Cinemachine.

If you have additional libraries that you want to use, [inform us](#) so we can make sure that it's compatible with the rest of the project.

Project Settings

If you are unsure how to set up your project with the correct settings, see the [How to set up your project](#) section. If during development you realize that you need to change additional project settings, please [consult with us](#) first. Keep in mind that build settings are not project settings and can be configured to whatever you need.

Basic Folder Structure

- Imports
- YourMinigameName
 - Art
 - Audio
 - Materials
 - Prefabs
 - Textures
 - Code
 - Scripts
 - Shaders
 - Scenes

The provided folder structure is a foundation you can build on. So you can customize the existing structure to fit your needs. Please keep in mind that for future students looking at your project, a clear folder structure can help them understand and navigate your project a lot quicker.

The only important distinction is the “Imports” and “YourMiniGameName” folders. The “Imports” folder contains all imported asset packs, and the “YourMiniGameName” folder should contain everything relevant to your game. The imports folder is provided and should generally be the same for all minigames. **Don't edit anything inside the “Imports” folder directly, always make a copy into the “YourMiniGameName” folder first!**

The goal is that one can take the “YourMiniGameName” folder and copy-paste it into another unity project together with other minigames, and have a single “Imports” folder that all minigames collectively use.

Scenes

The following scenes (with these exact names) must exist within your game:

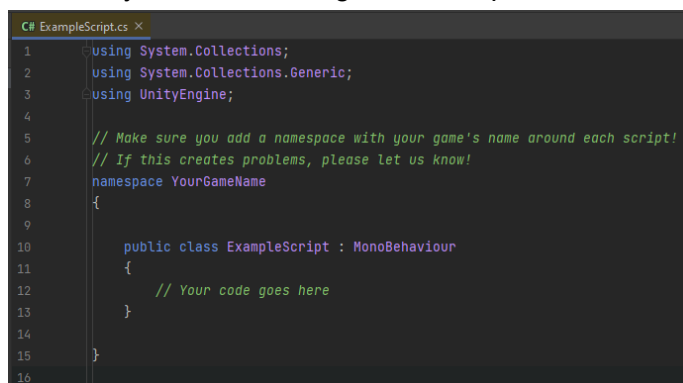
- **MainMenu** - Scene with the main menu. From here one can start your game (= load the MainGame scene), or look at your tutorial (= load the Tutorial scene).
- **Tutorial** - Scene with an explanation of your game. From here one can start your game (= load the MainGame scene). Depending on your game, explanations about button layout, objectives and rules of your game should be present.
- **MainGame** - Scene that starts your game logic when loaded. If you have multiple levels, you can have them in different scenes, but there still needs to be the MainGame scene as an entry point, from which e.g random level scene is automatically loaded.
- **GameOver** - Scene which gets loaded after your game finishes. Here an option to “Play Again” and an option to “Return to Menu” need to exist. Additionally, you can display a ranking, score or the winner(s).

The design and any additional content is up to you, as long as the basic functions as described above exist.

Scripts

You can create and name scripts/classes however you want, as long as you **make sure that all of your code lives inside a unique namespace.**

The name of your namespace is ideally the name of your minigame, but you can name it however you want, as long as it's unique.



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 // Make sure you add a namespace with your game's name around each script!
6 // If this creates problems, please let us know!
7 namespace YourGameName
8 {
9
10     public class ExampleScript : MonoBehaviour
11     {
12         // Your code goes here
13     }
14
15 }
16
```

If this creates problems, f.ex. when you need to work in specific unity namespaces, [contact us](#) and we'll help you solve the issue.

Assets

If you use the template, the available assets should already be imported and set up. If not, see the [How to set up your project](#) section.

Check out the demo scenes that show you what can be made with these beautiful assets. Demo scenes are usually located within the asset pack folder under the “Scenes” or “Demo” folders.

All assets are from the popular asset studio “Synty”. They are all in the same style and can be used interchangeably.

Check out their work here: <https://assetstore.unity.com/publishers/5217>

We currently own the following asset packs:

- Sci-Fi Space:
<https://assetstore.unity.com/packages/3d/environments/sci-fi/polygon-sci-fi-space-low-poly-3d-art-by-synty-138857>
- Town:
<https://assetstore.unity.com/packages/3d/environments/urban/polygon-town-pack-low-poly-3d-art-by-synty-121115>
- Prototype:
<https://assetstore.unity.com/packages/3d/props/exterior/polygon-prototype-low-poly-3d-art-by-synty-137126>
- Pirates:
<https://assetstore.unity.com/packages/3d/environments/historic/polygon-pirates-low-poly-3d-art-by-synty-92579>
- Fantasy Rivals:
<https://assetstore.unity.com/packages/3d/characters/humanoids/fantasy/polygon-fantasy-rivals-low-poly-3d-art-by-synty-118399>
- Farm:
<https://assetstore.unity.com/packages/3d/environments/industrial/polygon-farm-low-poly-3d-art-by-synty-146192>
- Fantasy Characters:
<https://assetstore.unity.com/packages/3d/characters/humanoids/fantasy/polygon-fantasy-characters-low-poly-3d-art-by-synty-97186>
- Icons:
<https://assetstore.unity.com/packages/3d/gui/polygon-icons-pack-low-poly-3d-art-by-synty-202117>
- Office:
<https://assetstore.unity.com/packages/3d/props/interior/polygon-office-low-poly-3d-art-by-synty-159492>
- Mini Fantasy:
<https://assetstore.unity.com/packages/3d/props/exterior/polygon-mini-fantasy-pack-96800>

- Mini Fantasy Characters:
<https://assetstore.unity.com/packages/3d/characters/humanoids/fantasy/polygon-mini-fantasy-character-pack-122084>
- Explorer:
<https://assetstore.unity.com/packages/3d/characters/humanoids/humans/polygon-explorer-low-poly-3d-art-by-synt-128996>
- City Characters:
<https://assetstore.unity.com/packages/3d/characters/humanoids/humans/polygon-city-characters-low-poly-3d-art-by-synt-106757>
- Spy Kit:
<https://assetstore.unity.com/packages/3d/vehicles/polygon-spy-kit-low-poly-3d-art-by-synt-143396>
- Snow Kit:
<https://assetstore.unity.com/packages/3d/characters/polygon-snow-kit-low-poly-3d-art-by-synt-134501>
- City Zombies:
<https://assetstore.unity.com/packages/3d/characters/humanoids/fantasy/polygon-city-zombies-low-poly-3d-art-by-synt-131930>

They have tons more, and if you find an asset pack from them that you want to use, you can [contact us](#) and the FH might buy it and add it to the collection!

Please try and stick to these provided assets first, however. This ensures a similar look between all minigames and also prevents massive, bloated asset packs that add tons of size to the project.

NOTE: Be careful about not changing the imported asset prefabs directly! Always make a copy first and modify that. For example, if you have a player model that you want to use, don't directly drag it into your scene from the imports folder, because you might add scripts to it and accidentally apply your changes back to the root prefab in the Imports folder.

Game Design

Game Type	Minigame
Duration	Around 1 - 3min
Multiplayer	local
Players	2 - 4
Modes	All Modes (1v3, FFA, 2v2, etc) are possible

General

The game that you should create is a minigame similar to Mario Party minigames. The goal is for it to be a fun, social couch game that people from every age and skill level can enjoy. It does not need to be highly complex or have tons of content. A simple, easy to understand game with clear rules and behaviors is what you should be aiming for. You can check out existing Mario Party minigames for inspiration!

Players

The minigame should be playable with 2, 3 and 4 players. So your game needs to support different amounts of players. For example, a game that is by its nature only playable with 4 active players would not work, since it does not support only 2 active players playing.

This needs to be considered when designing your game, since certain game ideas don't work with different player amounts. A 1v3 game is also a 1v1 game for example if there are only 2 active players.

How to set up your project

You should be able to download a template project from this link

<https://git.technikum-wien.at/innolab/summiteer/minigame-template>

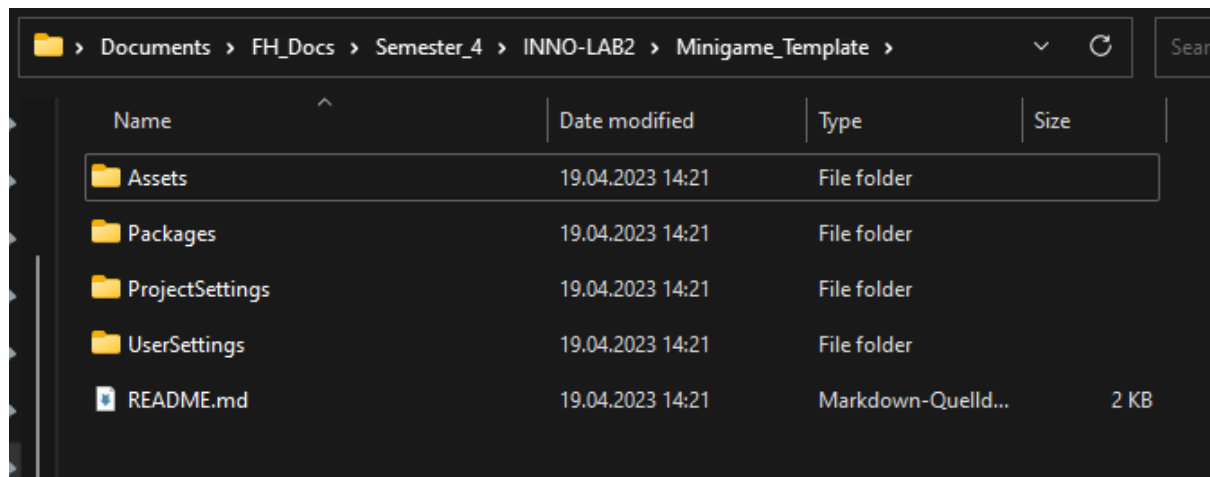
where everything has been set up and imported correctly. Keep in mind that you need to be in the FH network to access this repository (VPN).

This template can be used, or the project can be set up manually.

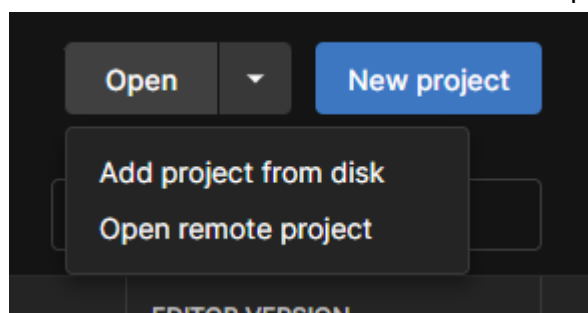
If you want to use the template,

1. Clone the repo and delete the “.git” items.

The files should then be in a folder with the name of your minigame. It should look something like this:



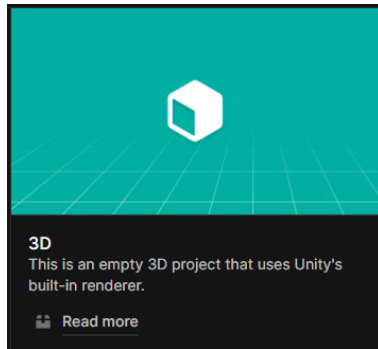
2. This folder can then be added and opened in Unity-Hub as a “project from disk”.



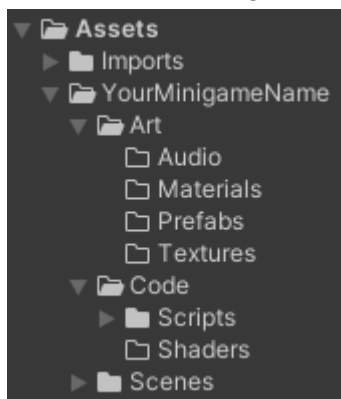
3. If the “Missing Editor Version” error shows up,
 - a. If you have a Unity Version 2021.3.XX already installed, you should be able to open it with your already installed version without major issues.
 - b. If you don't have Unity installed, or your version is too high, then install the missing Version. Make sure you select the option to add Windows Platform Build Support to be able to build your project.

If you want to create your own blank project, or the template is not working, be aware of the following:

1. Use the regular “3D” Template to create a new empty project. This sets up the render pipeline correctly.

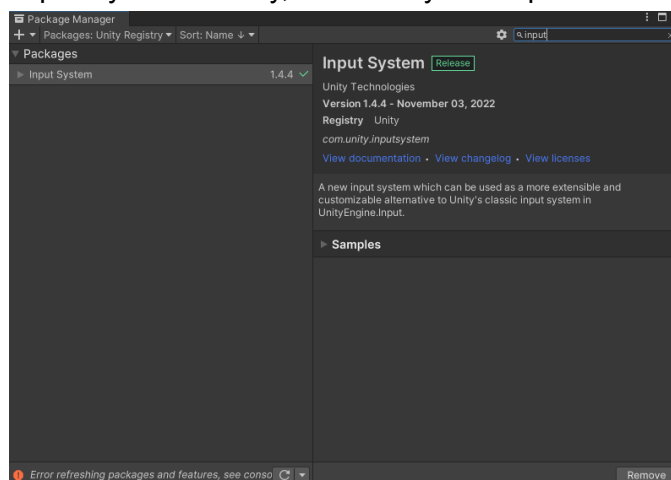


2. Create the following folders:



For more information, see the Section [Basic Folder Structure](#).

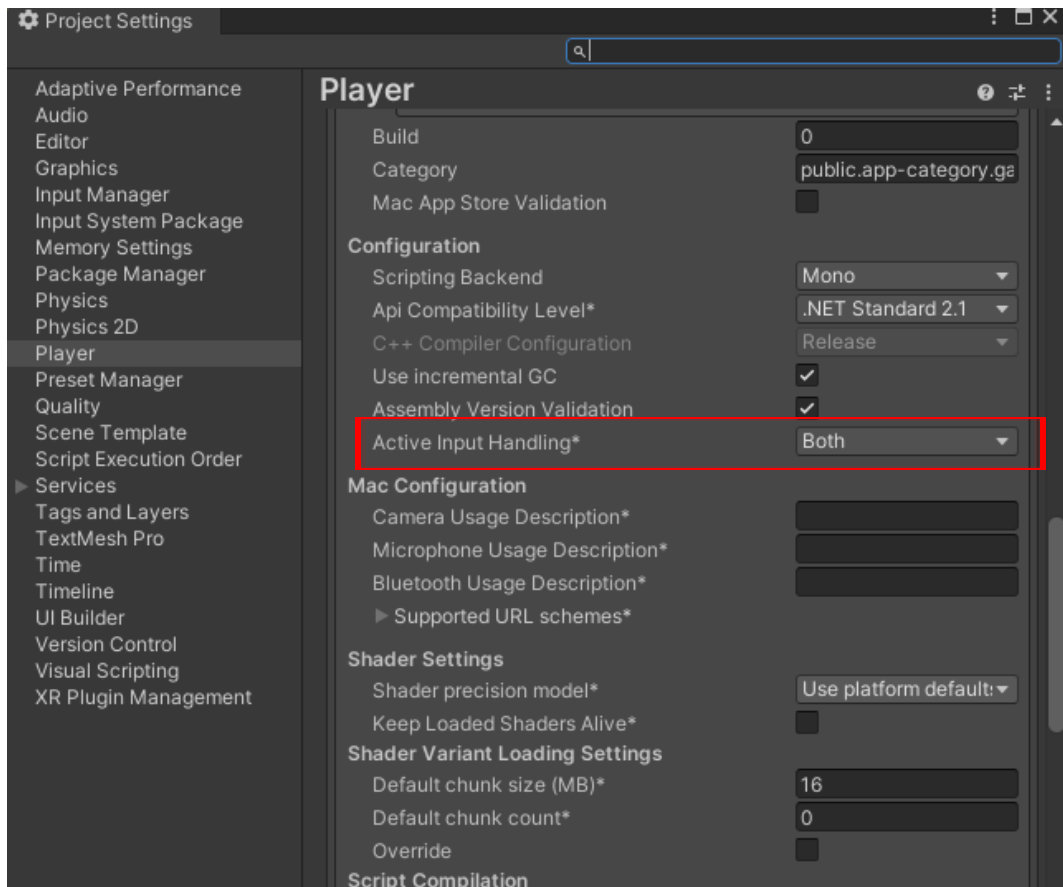
3. Install the “Input System” library from the Unity Package Manager. After installing the “Input System” library, it will ask you to update Settings. **Click “No”**.



If you clicked “Yes” by accident, it’s not a problem, since you’ll just update the automatic setting it did to a different setting in the next step.

If necessary, see the [Supported Libraries](#) Section for more information.

4. Go to “Project Settings” => “Player” => “Active Input Handling” and set it to “Both”.



5. Create 4 Scenes in your “Scenes” Folder named the following:

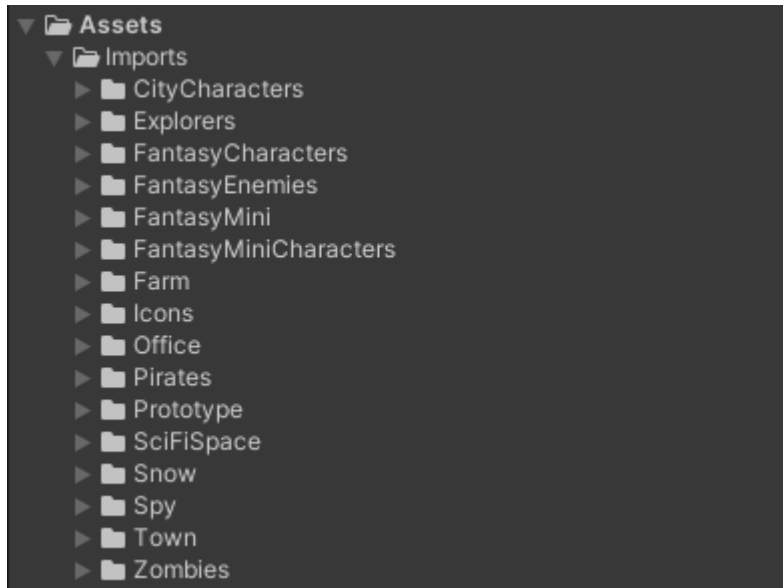


6. For more information about what each Scene should contain, see the [Scenes](#) Section.
7. You can either import the FH Asset Packs separately or import them all in bulk. You might also import all Asset Packs into a separate Project and only import the Assets you actually need into your main project.

You should have access to all the “.unitypackage” files, which you can just drag into your project hierarchy to import.

Important: If you import them separately, it creates a folder directly inside the Root “Assets” Folder. Please move all Asset-Pack folders into the “Imports”-folder like shown in the screenshot below. Otherwise your Project Hierarchy will become

cluttered very fast, and later students looking at your project might not know which folders are just imported asset packs and which are your own.



Checklist

- ☐ The project uses a unity 2021.3 version
- ☐ The project uses the default render pipeline
- ☐ The project has a clear folder structure
- ☐ The "Imports" folder only contains unedited asset packs
- ☐ The "YourMiniGameName" folder contains everything else
- ☐ The project has a scene called "MainMenu"
- ☐ The project has a scene called "Tutorial"
- ☐ The project has a scene called "MainGame"
- ☐ The project has a scene called "GameOver"
- ☐ The MainMenu scene can load the MainGame scene
- ☐ The Tutorial scene contains an explanation of the game
- ☐ The Tutorial scene can load the MainGame scene
- ☐ The MainGame scene starts the game logic when loaded
- ☐ The GameOver scene can load the MainGame scene ("Play Again")
- ☐ The GameOver scene can load the MainMenu scene ("Return to Menu")
- ☐ Imported asset prefabs are unchanged
- ☐ The project can be built & run on windows machines
- ☐ All code lives inside a unique namespace
- ☐ The "Input System" library is used to handle player input
- ☐ The game is playable with gamepad(s)
- ☐ The game supports 2 - 4 active players

Contact information

Here you can find the Mail and Discord handles of all main project members.

Project Supervisor:	Markus Petz	petzm@technikum-wien.at Petz#2391
Teamlead:	Samuel Muskovich	if21b157@technikum-wien.at Talimon#0843
Team members:	Stefan Kellner	if21b040@technikum-wien.at StefanK#6258
	Felix Kainz	if21b230@technikum-wien.at wisdom#0001
	Si Si Sun	if21b102@technikum-wien.at susu#1161
	Raoul Wograndl	if21b142@technikum-wien.at Raoul#3069