

Project: Minimum Viable Game Demo

98-127 Introduction to Game Development in Unity

<https://www.gamecreation.org/course>

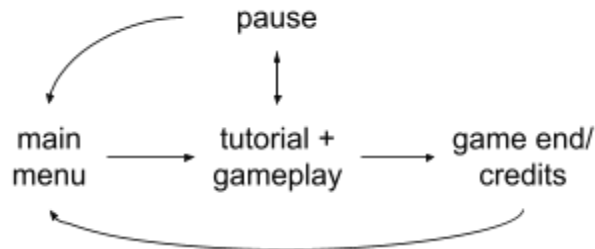
CHECK IN: Week of 23 Apr, 2026

PLAYTEST DUE: 11:59pm, 17 Apr, 2026

FINAL DUE: 11:59pm, 24 Apr, 2026

Description

It's time to see how everything you've learned can come together in a complete experience! In this project, you'll create a small but fully playable game demo that runs start-to-finish as a standalone build. Your demo should include a main menu, a tutorial/gameplay phase, a pause state, and a game end/credits sequence. These parts should flow together as follows:



To meet minimum requirements for full credit, we recommend you to implement a basic obstacle course for gameplay while meeting all relevant rubric criteria. However, we welcome you to come up with your own game ideas, and pursuing your own idea will give you extra credit. If you choose to do so, you can hash out the details with the instructors during check in.

Instructions

- **Check in:** This is an informal meeting with an instructor to discuss your goals for the project, as well as an opportunity to clarify any concerns you may have. The instructors will reach out to you to schedule a meeting when the time comes. Be prepared to come with an outline of your gameplay and choice of features to implement.
- **Submission:** You will submit a build of your game (Windows and MacOS) to the corresponding channel in the course discord as a google drive link (or anything that allows us to access the executable).
- **Playtest:** Those that submit a version of their build before the playtest deadline will receive a preliminary grade and feedback by the end of the weekend, providing an opportunity to fix the lacking portions of the project before the final deadline.
- **Late-work policy:** Note that the last day of class is a **hard deadline**. We will not grade any work submitted after Friday, 24 Apr. We recommend you to submit a build for playtest regardless of if you've finished your project at that point.

Rubric

Criteria		Grade
Check in	Meet with an instructor to discuss your plans and goals for the project	2
Project	Gameplay loop (as presented in description) is fully implemented	5
	Implement features from the list below. Each * is worth 5 points. 5 * are needed for full credit and up to 4 additional * can be counted for extra credit	25 + 20
	Additional grading details can be discussed for custom projects that go beyond requirements listed in this writeup	
Total:		32 + 20

Features

The following is a non-exhaustive list of features you could implement for your project. If you'd prefer to implement your own ideas, their * worth can be discussed during checkin.

Grade	Task	Details
*	Tutorial	Implement an in-game tutorial that teaches the player the core controls (for example, text prompts that appear at specific locations or events). If you choose not to build an interactive tutorial, you must still include at least a screen in your demo that lists all game controls.
	Consistent Animations and Sounds	Add animations and sound effects to all game elements that would reasonably have them (for example, UI interactions or additional entities you introduce), to the point where the player does not feel that a particular element is missing animation or sound.
	Acceleration for Player Controller	Add acceleration to the player's movement, smoothing both position and rotation changes.
	Jump Refinements	Add jump cut, coyote time and input buffer.
	Add VFX	Examples include small dust clouds when jumping or landing, fire, rain, or ambient particles floating in the air. However effects added should feel consistent and not like it was added just for the sake of completing this task.
**	Procedural Animation	Use a constraint from the animation rigging package to make the character respond dynamically to the environment (for example, having the character look at a target).

	Slope Interaction	Modulate player speed on slopes as per method introduced in slides
	Stair Interaction	Implement the floating collider method for stair interaction as introduced in the slides.
	Custom Shader	Use Shader Graph to create a custom shader and apply it somewhere in the scene. There are many options; examples include toon shader, water, or foliage.
***	Custom Procedural Animation	Create a custom constraint that has complexity comparable to the more advanced constraints in the animation rigging package. One recommended example is spring bones, or jiggle chain as introduced in the unity video linked in slides.
	Stair with Estimated Slope	If you implement additional slope estimation for stairs, ** Stair Interaction counts as this feature instead
*** *	Enemy AI	Use behaviour graph to implement behavior for an enemy. Since this would likely involve more than just the graph itself—such as setting up a NavMesh, triggers, and possibly additional art or effects—this feature is weighted significantly higher than most other features.