

CPSC 427 - Video Game Programming

Milestone 3: Advanced Game

For this milestone, you should have a complete playable game, and continue to support **all required** features from prior milestones (please fix any latent bugs). You should also include advanced features such as detailed geometry, non-linear motion, and time-stepping based physics. Test the playability of all new features and ensure alignment with your team's game development plan.

You should support robust continuous play with no memory leaks, crashes or glitches. Ask people outside your team to test play the game and modify your gameplay and interface as necessary based on their input.

(60%) Mandatory Requirements:

- **Playability (15%):**
 - Sustain *progressive, non-repetitive* gameplay for **5 min or more** that integrates all the new features (*with minimal oral instruction*). During the 5 min, the player should be able to interact with the game and see **new** content for most of the time.
- **Robustness (25%):**
 - Include proper memory management (no memory hoarding or leaks, the game should not hog memory even after extended play time) (10%).
 - The game should robustly handle *any* user input. Unexpected inputs or environment settings should be correctly handled and reported, and not crash the game (5%).
 - The gameplay should be real-time (no lag). Use a profiler to locate runtime bottlenecks and resolve them once located (10%).
- **Stability (20%):**
 - Include fully completed and playable prior-milestone implementations. Fix all bugs identified in prior marking sessions.
 - The game resolution and aspect ratio are consistent across different machines/displays.
 - The game code should support continuing execution and graceful termination, with no crashes, glitches, or other unpredictable behavior.

(40%) Creative Components: To obtain full marks you should implement a subset of the advanced features below. **Correctly** implementing multiple features can bring your grade above 100%.

- **Reloadability** (10%): The game should allow for full state saving for play “reload”. Users should be able to exit the game and restart at the same place, level, or chapter they left the game at, with all environment variables reset to the state they were in at save time.
- **Physics-Based Animation** (30%): Implement time-stepping based physical simulations which can either serve as background effects (e.g. water, smoke implemented using particles) or as active game elements (throwing a ball, swinging a rope, etc.). A subset of the game entities (main or background) should possess non-trivial physics properties such as momentum (linear or angular) and acceleration, and act based on those.
- **Complex Geometry** (20%): Incorporate one or more complex polygonal geometric assets. Implement an accurate and efficient collision detection method that supports this and other moving assets (include multiple moving assets that necessitate collision checks).
- **Complex Prescribed Motion** (10%): Use geometric splines (Hermite, Lagrange, Bezier, etc.) to implement smooth non-linear motion of one or more assets or characters. An example of a curve controlled animation is shown at <https://docs.unity3d.com/uploads/Main/AnimationEditorBouncingCube.gif>
- **Other:** As an alternative to the above you can implement a selection of basic (10%) or advanced (20%) features listed in the **MilestoneSubmissionForm.pdf** which were not part of prior milestones.

Grading here will necessarily be subjective: more complex features or those better fitting into the overall game will be rewarded with more points.

Note: You will receive full credit for any of the features above only if they are **fully** operational. You will receive points for creative components only if the mandatory ones are fully operational. Points will be deducted for buggy and/or incomplete implementations.

Documentation:

- Provide a README.md providing entry points to each of the implemented features and explain them where necessary.
- **Your submission should align with your proposed development plan:** Provide a write-up explaining how your milestone aligns with the plan. Explain all discrepancies and **submit an updated proposal** when such discrepancies occur.
- **Game Design Documentation:** Document the ECS design pattern used in your game. Enumerate the game entities and actionable components used. Draw a diagram of the

interaction between entities and components. *Highlight any changes versus the previous milestone.*

- Please submit a filled **MilestoneSubmissionForm.pdf** with this and all subsequent milestones.

Submission: Submit the code and associated documents using the course Git repository that has been set up for your team at <https://github.students.cs.ubc.ca/CPSC427/team#>. The repository is hosted on the UBC servers and will be accessible only to enrolled students. *Note that each team member is also expected to submit their individual progress & feedback report via 'handin'.*