CPSC 427 - Video Game Programming

Fall 2019/20

Milestone 3: Advanced Game – November 8, 2019

For this milestone, you should have a complete playable game, and continue to support all features from prior milestones. 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 the 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%) *Milestone requirement:*

- **Playability** (10%):
    - 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) (15%).
    - 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 complete playable prior-milestone implementation. Fix all bugs identified in prior marking sessions.
    - The game should be consistently playable across different machines/displays.
    - The game code should support continuing execution and graceful termination, with no crashes, glitches, or other unpredictable behavior.

(40%) *Optional Components*: To obtain full marks you should implement a subset of the advanced features below. **Correctly** implementing multiple features can bring your grade to 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 they left the game, with all environment variables reset to the state they were in at save time.
- **Physics-Based Animation** (30%): Implement time stepping based physical simulation which can either serve as a 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

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

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.

Submission: Submit the code and associated documents using the course Git repository that's been set up for you 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'.*