M3: Production Release (Game)

Overview

As a team, you will complete and deliver a ***production release*** (sometimes referred to as a ***gold master*** in the game industry), implementing all game functionality, as your final deliverable. As the name implies, the production release must be a complete product with all final features integrated, functional, and polished. All character types (player and non-player) must be completed, and all mechanics. Final animations, sound, and subsystems must be complete and integrated into the game. This includes tweaking of inventory systems, behaviors of artificial intelligence, and levels / stages. The gold master is the final version of a game that is delivered to end users, and as such it should be usable by a typical player. The production release should represent approximately 100 hours of effort ***per person***, cumulatively (including time spent on the prototype).

Specification

The deliverable should meet the following requirements.

User Experience

The user feedback system, including UI and sensory response elements, should complete and polished in the production release. All elements of user control should be completed.

Interface

All user interface elements should be visible, and usable, and polished. Options should be intuitive and consistent. Additionally, the currently selected button(s), menu option(s), or other dialog element(s) should be highlighted. General status changes must be visible in the Heads-Up Display (HUD).

Navigation

All user mechanics should be functional and polished. Controls should be thoroughly tested. There should be no bugs. Controls should not require significant time for acclimation; they should be predictable and easy to use. New features and character abilities should be introduced throughout the game in a manner that allows the player to integrate them into the experience seamlessly. They should be well explained. The team should avoid clumping features together in a way that inhibits effective player progression.

User Perception

The game must be responsive, and controls must be intuitive to users. Teams should have solicited user testing from non-team members to ensure general usability. Users should report an enjoyable experience. Game difficulty should adjust with new players as those players learn how to play the game. This should involve minimal *frustration* on the part of the player. Ineffective difficulty pacing – too fast or too slow – takes away from the play experience and should be avoided.

Responsiveness

Interactions between entities in-game must be indicated by the sensory experience (visually, audibly, and/or via haptic feedback). Interactions include, but are not limited to, collisions, attacks, or discourse. Changes in character state must also be indicated via the characters themselves. (For example, a character may glow red when damage is taken, and a heart monitor sound may become audible when health falls below a threshold.) Additionally, reaching game-level milestones must be indicated (such as by displaying a flag on the terrain).

Build Quality

Projects receiving a perfect build score will have no detectable bugs. All content, including art and code systems, should be present and polished.

Robustness

There should be no crashes or glitches in the production release of the project.

Consistency

Except where explicitly by design, the game should act predictably; i.e., for the same input / use case, it should yield the same result. When / if the game behaves unpredictably, it must be for a clear and compelling reason.

Aesthetic Rigor

No cosmetic issues should be present. Under no circumstances should aesthetic issues cause the game to be unplayable. All assets should be present and functional within the context of the game. Assets and their integration within the game should be polished.

Features

All freatures should be complete, with edge cases tested and accounted for, and with failure modes anticipated and handled.

Front-End

For each game feature, control scheme(s) must be implemented, usable, and smooth (including player mechanics), and the game state must be reflected on-screen. Changes must connect to the *persistent state*.

Persistent State

For each game feature, the release should make proper use of data structures and update to the game state. This must connect to the *front-end* and the *back-end*.

Back-End

For each game feature, all processing (e.g., physics updates, video update, and other calculations) must be implemented. This must connect to the *persistent state*.

Submissions

Your submission will be a tgz (gzipped tar) archive including the following:

- Source code and assets for submission
- README file

Source & Assets

Submissions should include all source code, including build configurations and scripts (e.g., setup.py and/or Cargo.toml) and run scripts as necessary. The project should be runnable using an executable command without parameters and should be prepared by build scripts. For server applications, a command to initialize the server state for first run may be included. Submissions should also include all pre-built assets necessary for application function (e.g., pre-built database collections necessary for initial function and/or visual/audio assets).

README File

A README file, either in plaintext or markdown (UTF-8) should be included. The README should have, at a minimum, the following information:

- A description of the project
- One-line build command (**desktop and mobile**) and installation package executable link (**desktop**)
- Link to the repository for the project (e.g., GitHub) [Note: if private, you must add instructors]
- Link to installation package executable (desktop) or link to the application package (APK) (**mobile**)
- Run executable command