UF UNIVERSITY of FLORIDA
Herbert Wertheim
College of Engineering

M2: Design Prototype

Overview

As a team, you will complete and deliver a ***design prototype*** demonstrating the feasibility and core features of your project. The features demonstrated will encompass a ***vertical slice*** – i.e., an iteration of your project that has all core features integrated in at least one dimension, though it need not be polished. The prototype should not involve implementations of all aspects of every feature; instead, the design prototype is intended to provide a basis for architecture and practical testing (including usability and user experience). The goal of the design prototype deliverable is to identify exactly how the project will be structured and to provide a foundation for the second phase of the project. The prototype should represent approximately 30 hours of effort ***per person***.

Specification

The deliverable should meet the following requirements.

User Experience

The user feedback system, including UI and sensory response elements, should be in place by this milestone. All elements of user control (including player character rigging and actions) should be present.

Interface

All user interface elements should be visible and usable. 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. Controls should be thoroughly tested. Bugs should be minimal, if present, and must not detract significantly from the overall experience. 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
This game build should endeavor to avoid major bugs. All content, including art and code systems, should be present.

Robustness
By this milestone, crashes should not occur within the context of regular (unexceptional) play. There should also be no major or noticeable glitches within regular play.

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 major cosmetic issues should be present. Any aesthetic issues that are present should not cause the game to be unplayable. All assets should be present and functional within the context of the game.

Vertical Features
All major architectural elements should be complete with at least one demonstrative implementation.

Front-End
For each major game feature, at least one variant of the control scheme must be implemented and usable (including ALL player mechanics), and the game state must be reflected on-screen. Changes must connect to the *persistent state*.

Persistent State
For each major game feature, there must be a demonstration of 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 major game feature, at least one variant of the 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