CS4455/CS6457 Final Project Grading Criteria

Required Gameplay Concepts:

Note that the sub-list items below are not meant as exhaustive checklists, but instead demonstrate examples of the types of things we are looking for in each assessment category. Also, you will see that there is overlap between some of the categories. Be especially mindful of these similar objectives that keep reappearing as they are generally the most important for the project.

It must be a 3D Game Feel game! (**5 pts**)

- You have implemented a 3D Game that can be defined as a Game Feel game
- Achievable objective/goal? (E.g. player can complete, or alternatively fail at, a level. NOT a sandbox)
- Communication of success or failure to player!
- Start menu
- Able to reset and replay on success or failure (e.g. In Minecraft when "You died", there is a "respawn" button)
- Sorry, no first-person perspective games (e.g. FPS).

Precursors to Fun Gameplay (**20 pts**)

- Goals/sub-goals effectively communicated to player
- Your game must provide interesting choices to the player
- Your player choices must have consequences (dilemmas)
- Player choices engage the player with the game world (e.g. buttons, climbable ledges, gun turrets, computer terminals, vehicles, etc.) and inhabitants (AI-controlled agents)
- Avoid Hollow, Obvious, and Uninformed Decisions
- Avoid Fun Killers (micromanagement, stagnation, insurmountable obstacles, arbitrary events, etc.)
- Your game must achieve balance (balance of resources, strategies, etc., as appropriate)
- Reward successes, and (threat of) punishment for failure
- In-game learning/training opportunities (e.g. gate restrict player to safe area until basic proficiency in new skill is demonstrated, etc.)
- Appropriate progression of difficulty
- Avoid opportunities for players to trivially beat your game challenges. That means playtesting to observe emergent strategies that might break your game!

Skeletal-Animated 3D Mesh Character Controller with Real-Time Control (**20 pts**)

- Based on skills learned in Milestone 1
- Mecanim-controlled, Blendtree-enabled, player-controlled character
- Note: Character turning can be tricky when exclusively handled via blended animation, particularly if you aren't the author of your set of turning animations and can't easily tweak them as needed. Therefore, for the alpha/final project you may elect to be selective in the use of programmatic turning and root motion animated turns (perhaps blend the two). Turning in place can be entirely programmatic if your gameplay necessitates the precise control this approach affords.
- Player has direct, dynamic, variable/analog-style control of character movement majority of time (ideally support a popular analog controller like PS4) Examples: analog stick movement, Mario's variable jump height from button hold duration, modifier-button control of a speed boost
- Choice of controls is intuitive and appropriate (e.g. shouldn't make difficult button mappings or controls where it is unnecessarily difficult to interact)
- Fluid animation. Continuity of motion (no instantaneous pose changes, no glitching, unintended teleporting, character stuck floating in air, etc.)
- Low latency responsiveness to control inputs
- Camera is smooth, always shows player and obstacles in a useful way, automated follow or intuitively controllable as appropriate
- Auditory feedback on character state and actions (e.g. footsteps, landing, collisions)

3D World with Physics and Spatial Simulation (**20 pts**)

- Based on skills learned in Milestone 2
- You have synthesized a new environment rather than just use a purchased level from the asset store. You can certainly make use of purchased assets, architecture, terrain, etc., but you *must* synthesize a unique environment tailored to your character's abilities and the challenges you wish to support.
- Both graphically *and auditorily* represented physical interactions
- Graphics aligned with physics representation (e.g. minimal clipping through objects)
- Appropriate boundaries to confine player to playable space (players should never encounter avatar falling to negative infinity or escaping the level; use invisible colliders and death zones when necessary)
- A variety of environmental physical interactions including:
    - Interactive scripted objects (buttons that open doors, pressure plates, computer terminals, etc.)
    - Simulated Newtonian physics rigid body objects (crates, boulders, detritus, etc.)
    - Animated objects using Mecanim (can be used for non-humanoid animations too!), programmatic pose changes, kinematic rigid

body settings, etc. (moving platforms, machinery gears, gun turrets, etc.)
    - State changing or destroyable objects (glass pane that shatters, boulder that breaks into bits, bomb that blows up, etc.)
- 3D simulation with six degrees of freedom movement of rigid bodies
- Interactive environment
- Consistent spatial simulation throughout (e.g. Running speed remains same regardless of framerate. gravity constrains maximum jump distance; shouldn't be able to jump more or less in different cases unless obvious input control or environmental changes are presented to the user).

Real-time NPC Steering Behaviors / Artificial Intelligence (**20 pts**)

- Based on skills learned in Milestone 3
- Multiple AI states of behavior (e.g. idle, patrol, pathfinding, maniacal laughter, attack-melee, attack-ranged, retreat, reload, flossing, etc.)
- Smooth steering/locomotion of NPCs.
- Predominantly root motion for humanoid characters. (See Mecanim character control discussion above)
- Reasonably effective and believable AI?
- Fluid animation. Continuity of motion (no instantaneous pose changes, no glitching, unintended teleporting, character stuck floating in air, etc.)
- Sensory feedback of AI state? (e.g. animation, facial expression, dialog/*sounds*, and/or thought bubbles, etc., identify passive or aggressive AI)
- Difficulty of facing the enemy is appropriate?
- Perceived as a *fair* opponent? You may need to implement a perceptual model for your AI.
- AI interacts with and takes advantage of environment? (presses buttons, takes cover from attack, collects resources, knocks over rigid body objects, etc.)

Polish (**15 pts**)

- Partially based on Milestone 4 (as well as other milestones and project work)
- Overall UI
    - Your software should feel like a game from start of execution to the end
    - There should be no debug output visible (you can remove your team name from the heads-up display for alpha and final project)
    - Ability to exit software at any time
    - No "test-mode"/"god-mode" buttons, etc. (OK if accessible via menu option or unlikely-to-press-accidentally button combo like: ↑,↑,↓,↓,←,→,←,→,B,A,SELECT,START)
    - GUI elements should be styled appropriately and consistently

    - Transitions between scenes should be done aesthetically (e.g. fade in, fade out, panning cameras, etc.)
- Environment Acknowledges Player
    - Note: there is obvious overlap here with the Spatial Simulation category above. In regards to polish, we are looking for environment interaction enrichment/appeal, as opposed to impact on core gameplay.
    - Should include many of the following:
        - Scripted aesthetic animations (swaying trees, grass, bugs under rocks, etc.)
        - Proximity-based events (alien plants that retract if you get near, picture frame falls off wall, etc.)
        - Surface effects, such as texture changes or decals (e.g. dent-able surface, bullet holes, etc.)
        - Particle effects (e.g. dust or splashes around footsteps)
        - Auditory representation of every observable game event
        - Physics event-based feedback such as particle effects and one-shot audio for collisions of different types
        - Sonficiations that map physical properties to audio effects. Example: Wheeled robot gear whine generated via attenuation and frequency shift of looped audio as a function of robot's speed
- Cohesiveness / Unified Aesthetic
    - Artistic style (extremely simple is fine! But be consistent and thoughtful)
    - Shading and lighting style
    - Unified color palette throughout game
    - Sound theme, including consonance (also avoid monotonous effects)
- Appeal
    - No glitches
    - No easily escaping the confines of the game world (make proper and plausible barriers)
    - No getting the player stuck
    - Stable (e.g. should play consistently the same on variety of hardware)

Fun / Engaging / Immersiveness / Emotional Response / Gestalt / "Flow" Achieved (~ 10 pts max bonus points)

NOTE: This category reserved for the FINAL project and *not* the alpha.

This category is acknowledgement that your game is *greater than the sum of its parts* and is particularly fun and/or highly engaging! Earning points here means that the instructor(s) felt that your game achieved something special that is not accurately reflected by the criteria above. Assessment of this

category can only bring your grade UP (i.e. bonus points). That said, there is something almost mystical about making a really fun game and even the most competent and industrious group might make a game that doesn't reach this level of achievement. You still need to have a "Game Feel" style game to get acknowledgement in this category (along with the technical requirements), and rarely will a game that is significantly deficient in terms of the project requirements gain many "fun" bonus points.

Submission (Up to 20 points deducted if out of compliance)

- Your group showed up ON TIME and gave presentation?
- *Quiet and paying attention* during other team presentations (all presentation days)?
- Have the project deliverables been submitted properly to TSquare?
- Instructional **Readme** file?
- **Video** documenting game concept and play?
- **Playable game binary provided**?
- Copy of digital **presentation materials**?
- **Team Member Assessments** submitted?
- Optional: **Video/Demo Release** for sharing