Robust Game: Stealthy Sam

CPSC 436D - Video Game Programming

Spring 2018/19

Team Members:

Caroline Chang	20044146	d6g0b
Sam Chow	27453142	i6x9a
Mah-dir Sheikh	36718154	p4u0b
Sam Veloso	33140154	m6o0b
Sam Goldwax	13433157	u9k0b
Nicolas Gagnon	26528142	l4f0b

Milestone Requirements:

Sustain progressive, non-repetitive gameplay for 6min or more including all new features. The game should not terminate early and should allow infinite even if repetitive gameplay

- Player is capable of playing all four rooms, including the final boss room. The final boss room includes four cauldrons that can be illuminated with the player's torch. After successfully avoiding boss projectiles and lighting the cauldrons, the player is rewarded with a Victory Screen and has the option to restart the game. Throughout the gameplay, the player can choose to restart at any point or save their progress through our newly developed save-state functionality. Gameplay allows for more than 6 minutes of puzzling challenging and stealth based exploration.

Support real-time response rate (i.e. lag-free input).

- User inputs are read real-time without delay. Responses are interpreted on each tick and the corresponding actions occur without lag.

Include proper memory management (no excessive allocation or leaks). The game should not have any undefined behavior, memory leaks or random crashes. The game should not hog memory even after extended play time.

- Components have proper destructors and are passed through as pointers (instead of unnecessary pass-by-value copying). When switching rooms or restarting all the maps and vectors loop through and delete the pointers.

- There has been a significant improvement on performance when switching rooms and restarting the game. Memory usage has decreased significantly as well since we switched the components to pointers (in the systems).

The game should robustly handle any user input. Unexpected inputs or environment settings should be correctly handled and reported.

- With over 10 key commands and mouse clicking, our input management has been isolated in the InputSystem and correctly responds to user inputs. Collision checks are performed in real-time to help prevent unexpected responses.

The gameplay should be real-time (no lag). This included improving your collision handling using effective detection strategies. You should support dozens simultaneously moving main or background assets.

- Multiple assets are now checked for collision :
  - Enemies colliding with torches or Sam
  - Torch with assets or cauldrons
  - Sam colliding with assets, keys, closets, walls, torches and enemies
  - Boss missiles with assets or Sam.

- Collision are all performed efficiently and real-time

The game should allow for some form of state saving for play “reload”. Users should be able to pause and restart in a graceful (if not perfect) manner.

- The user can pause or unpause the game at any point while Sam is alive with the use of the Space bar. Only once paused, the user can then save game progress (keys collected and current room) with the ‘B’ key and load any saved states with the ‘N’ key. The user is loaded to the beginning of whichever respective room the user saved in and all assets (excluding keys) are re-initialized to their starting position.

The physical effects should be correctly integrated in time and should not be locked to the machine’s speed by correctly handling the simulation time step and integration.

- Each frame time is processed into smaller fixed time chunks (correctly integrated in time). These smaller fixed time chunks are used to update our game state rather than a potentially large frame time (locked to machine’s speed), which will prevent breaking collision/physics/movement.

Stable game code supporting continuing execution and graceful termination

- Our code can be continually executed without bugs and we are able to terminate the program without any problems.

Creative:

You should implement one or more additional creative elements. These can include additional assets, rendering effects, complex gameplay logic, or pre-emptive implementation of one or more features from subsequent milestones.

- We have implemented a light dimmer that makes the player's surroundings darker.
- Different light sources (torch, Player, cauldrons) interact with one another.
- Torches can been thrown in every cardinal direction, as well as diagonally, through mouse click.
- Ghost preferentially chase torches over Sam, and extinguish them after collision.
- Extensive sprites to match our original level designs.
- Refactored the original code to model the ECS design.
- Refactored shader to facilitate feature extension.
- Added additional interactable closets in each of the levels.
- Vision cones that follow the enemies.
- Interactable cauldrons that provide additional light sources for the user.
- Revamped the tutorial screens to include all the instructions at once; accessible throughout gameplay
- Background music has been added to reflect the appropriate spooky tones for the game, also sound effects when dying and picking up keys have been added.
- A boss that shoots missiles at sam when not running from lit torches has been added. These missiles kill Sam if they touch him.

Your submission should align with your proposed development plan: Provide a write-up explaining how your milestone aligns with the plan. Explain all discrepancies.

In our original plan, we had these features planned for completion for this milestone:
Week: March 29th - **Robust**

- World shader
  - We have this implemented, and our character successfully travels with and takes a hole in the world shader with him, provided he has a torch, torches and cauldrons penetrate the world shader. Enemy vision cones also appear in the shader.
- Start implementation on interactables
  - We have currently keys implemented, closets you can hide in, and torches you can use for interactables.
- Level layout
  - We have all four rooms created that we proposed.
  - We have implemented puzzles pieces and enemies in each of the rooms.
- Gameplay

    - Enemies follow a decision tree allowing for players to strategically use torches to distract enemies.

We made all of our major goals for the playability milestone.