COS 426 Final Project: *Late For Class!*
John Hart and Hitesha Ukey

1. *Abstract*

In this report, we will outline our 3D game titled *Late For Class!*, our "I Spy"-like first-person game that puts players in the place of a college student late for class, and the details surrounding the game. We will give a short introduction to what the game is and the related work in the field. We then will delve into how the game was implemented and our unique approach to the game idea. We will go into the methodology behind the game mechanics and what tools were leveraged to make them. We will discuss the success of the application, our opinions about the effectiveness of our approach, improvements that can be made, and the lessons learned during development. We will then outline how we felt the project went, what we can do in the future to better the application, and how each member of the team contributed to the project.

2. *Introduction*
   a. *Goal*

Our goal was to create an "I Spy" style game in a 3D modeled environment. The added context to the game is that the player is a student who is running late to class and needs to search their room quickly for a missing item. We chose the setting to be a dorm room and to set up the scenario of the player being late to class in order to add relevance to Princeton and strengthen our game concept. To that end, we aimed to create a 3D model of a typical dorm room, populated with common objects and furniture, and design a user interface to allow the player to search the room and select objects easily. We also aimed to add a time constraint to increase the difficulty of the game.

   b. *Previous Work*

We came up with the idea of *Late For Class!* through games like *Gary's Mod* minigame "Prop Hunt" and other games that leverage the idea of needing to find an object in a limited period of time. The way we wanted to interact with the idea, however, was through a hide-and-seek game with an inanimate object. In other words, we wanted to make users have to find a static part of the scene similar to "I Spy" as fast as possible in a time environment – we wanted to make an intense version of "I Spy" that combined its premise with an environment not so keen on playing along (hence the idea that the object would be "hiding" from the player.

   c. *Approach*

Our overall approach to build this game consisted of utilizing the 3D modeling library THREE.js for a framework to simulate the 3D environment, JavaScript to create the overall website, and external sources for specific object models/textures/audio. We implemented a movable camera to give the player the ability to look around the room, collision detection to simulate real life interactions with a 3D world, and texture mapping to allow for customization of the room design. We used JavaScript functions to handle events, keep track of time, edit html elements, and make the game interactive. We integrated 3D models from Sketchfab.com and used audio files from Freesound.org (see *Works Cited*). We used the technique of raycasting to identify objects selected by the player, which was synced with input events.

3. *Methodology*

a. Dorm Scene / Texture Mapping

To create the scene, we followed the general code structure provided in the starter code of this project. We extended the Scene class from THREE.js called DormScene to create an overall object holding all the elements of our dorm room scene. Within this class, we instantized all the dorm objects, furniture, and room elements (like the walls and the floor). In separate files, we extended the Scene class to create classes representing individual objects (closet, dresser, bed, desk, etc.) and position them in the room. These classes used the GTLF loader from THREE.js to import models as well as basic geometries and materials from THREE.js to build some elements from scratch. We also used the Texture loader from THREE.js to map our own textures onto objects using either MeshPhysicalMaterial or MeshPhongMaterial. All the objects have custom textures except for the laundry basket and the water bottle. To look around in and move within the scene, we used THREE.js's PointerLockControls combined with key inputs to adjust where the camera was looking and where the camera (player) position was. We also created collisions such that the player was contained within the playable area and that they could not phase through objects. To select objects in the scene, we utilized a ray tracer coming from the camera to detect what the closest object the player was looking at was.

b. Timer

The internal timer was implemented using the window.setTimeout() and window.clearTimeout() functions as well as the Date.now() function in JavaScript. The timer visible on the screen was continuously updated through the animation handling framework provided in the starter code. To make the timer seamless with every game pause and start initiated by the player, we kept track of every start time to accurately calculate the remaining time. We also synced the length of the timer with the length of the background music.

c. Overlay

The overlay elements, which are the start/pause/end screens, timer box, and object box, were all created using JavaScript functions designed to add and manipulate HTML elements and their CSS styling. To achieve the overlay effect and reduce stylistic changes from window resizing, we manipulated the CSS position type of each element and set its dimensions/position using percentages. We integrated code to update the HTML elements within event listening code to allow the screen elements to change in response to player input.

d. Sound

The background music and sound effects were implemented using THREE.js functions/objects. We used Audio, AudioListener, and AudioLoader classes to integrate the sounds we chose from online sources and sync the audio to the gameplay. We connected the sounds to the camera, and then played and paused the sounds according to keyboard events.

4. Results

To measure our success, we tested locally within the development team and received informal feedback from course staff and potential users. Due to the time constraints of the development period, our capacity to conduct a more formal user evaluation or experiment to deduce the playability and general feel of the application was severely limited. To work around this, we placed a focus on improving usability during development based on the feedback we were able to receive from users within our own personal circles. The application was made more

intuitive and user-friendly during each iteration of development, and we entered each iteration with feedback, albeit limited, to improve upon the previous version.

5. *Discussion*

a. *How promising was our approach?*

We believe the approach we took is not only functional but scalable. Throughout development, there was a focus on modularity and writing code that was forward-looking. For example, there was an emphasis on keeping any functions that altered a Player object's state or used its state to perform complex calculations inside the Player class to improve the clarity of the code and to build the foundation for possibly having multiple players in the scene at once. This both eased the process of making the game function as intended and the ability to add new features and functionality into the game with relative ease.

Of significant importance is also the relatively simple libraries we used. For example, we pushed ourselves throughout development to stay inside the ThreeJS community since adding new features would then not conflict with the functionality of the previous version. Keeping the outside library and code use simple means that future expansion of the application can be accomplished trivially without having to worry about certain libraries or sections of code conflicting with each other.

b. *How could the approach be improved?*

While the application is very scalable and understandable from a developer's perspective, we believe that more focus on features and functionality users want would greatly benefit the application. Since our capacity to get holistic feedback on the application was limited, we likely did not implement all of the characteristics and gameplay that users would have wanted. Considering the importance of user input to games, we feel that the approach could be improved by utilizing an improved iterative development approach that has more user input and feedback.

c. *What are the next steps of the application?*

The next steps would be as follows: clean up the look of the application, fine-tune the existing features and functionality, and add new features. The look and feel of the application can use improvement, and we feel that spending more time on making the environment more immersive for users would make the experience that much more fun and replayable. The existing functionality is enough for the game to be played given the rules, but there are bugs and issues that need to be ironed out so that the gameplay becomes a seamless experience for players. After these previous steps and improvements, we feel that the game would benefit from an expansion of features – randomization of levels and objects, simulated dynamics, etc. – that would give the game more uniqueness.

d. *What did we learn from the project?*

We learned about how to apply the skills we learned in the course, our own capabilities, and what it takes to make a 3D game. For us, the course material could feel at times disjointed and disconnected, as there are many moving parts to the whole picture of computer graphics. However, working on this project taught us that everything we have learned was like a puzzle piece – only by fitting each piece together could we obtain the full picture of a 3D game. Doing this led to discoveries about what we were capable of, even given the short time constraints of the project. We feel very proud with the work we have been able to complete on this project, and

are more confident in our capabilities to make or improve an application similar to the project. In the same vein of thought, however, we also discovered how difficult a project such as this can be and how much knowledge and effort needs to be put into it for it to succeed. Knowing this, we feel better prepared to take on a similar project in the future.

6. Conclusion

a. Success of our Implementation

Our project was successful in laying out the bare bones of the game concept, including the dorm room environment and player interaction. We were able to design a dorm room style scene, albeit with limited objects and furniture, with a customized layout and texture designs. We created a player/camera that can move through the room in all directions that simulates colliding with objects and can look in all directions paralleling human eyesight/range. We successfully identified objects in the room that the player is "looking" at using raycasting, allowing us to set up gameplay which determines whether the player won or lost the game. We were able to add lots of event handling functionalities for the player to interact with the scene (mostly through capabilities to move and look around) and add sounds to increase the game enjoyment for the player.

b. Current Issues

The weaknesses of our project lie in a lack of complexity and small limitations in functionality. We were unable to establish replayability because there is a single room environment and a single object to search for which has a static position. Even the scene we did implement could be improved upon, such as through customizing the outside environment around the box and adding a ceiling, in addition to adding many more objects and design elements to boost attractiveness and interest.

There are also some refinements that we were unable to prioritize, such as circumventing an issue arising from either THREE.js or the starting framework that causes errors when the player clicks around/interacts with the scene or game too quickly. Collisions could also be improved upon due to some slight bugs with the physics used to implement it – certain surfaces cause the player to "stick" to them, only allowing the player to leave by jumping.

c. Future Improvements

As mentioned above, one of our top priorities in improving the game would be to add complexity to the room scene. We would add more furniture and common objects to make the room look more realistic and make it more difficult to search for the missing object. We would also add other small missing objects so that the game can be played multiple times. Another improvement would be to place these missing objects in random locations.

A stretch goal we were not able to implement was interacting with static objects like furniture to look inside of them for missing objects. For example, the player could open and close the dresser, closet, or desk drawers. Similarly, adding simulated dynamics to the game (e.g. ragdoll physics to room objects) would have added some flavor to the game and allowed players to have a more chaotic experience (in a good way).

Another improvement could be to polish up the way the game presents itself, such as adding cutscenes at the beginning and end of the game. In the same vein of thought, making the sounds and music more specific to a college student running late to an important class would add to the atmosphere and environment the game creates.

7. *Contributions*
   - **Jack**
     - Handled camera controls and basic lateral player movement.
     - Handled player input, keybinds, and event listeners.
     - Made “fake” physics for jumping and sprinting.
     - Implemented player collisions with the environment.
     - Implemented basic ray tracer for selection of objects in the scene.
     - Confined player to room upon game start.
     - Made the first version of the pause menu.
     - Laid groundwork for implementation of a loading screen (not currently functioning).

   - **Hitesha**
     - Designed dorm room layout: lighting and objects
     - Integrated 3D models found online into scene
     - Applied texture mapping to relevant models
     - Added background music and sound effects
     - Added some html overlay elements like timer and object picture
     - Wrote code to keep track of time
     - Added start screen and win/lose end screens

8. *Works Cited*
   - 3D Models from Sketchfab.com
     - **Mattress:** This work is based on "Dorm Floor Mattress With Sheets [No Texture]" (https://sketchfab.com/3d-models/dorm-floor-mattress-with-sheets-no-texture-1fa7d1844949448b8b8abaf4183126c5) by Ingrid Bie (https://sketchfab.com/ingridbie) licensed under CC-BY-4.0 (http://creativecommons.org/licenses/by/4.0/)
     - **Desk:** This work is based on "Desk" (https://sketchfab.com/3d-models/desk-95fe0ea09f97465fb5183573b44f2b7b) by James (https://sketchfab.com/James_999) licensed under CC-BY-4.0 (http://creativecommons.org/licenses/by/4.0/)
     - **Chair:** This work is based on "CHAIR" (https://sketchfab.com/3d-models/chair-b07d263a7ab942e6935e77cd75bf1194) by vUv (https://sketchfab.com/vovaustimuk) licensed under CC-BY-4.0 (http://creativecommons.org/licenses/by/4.0/)
     - **Water Bottle:** This work is based on "Water Bottle" (https://sketchfab.com/3d-models/water-bottle-42f0a41639a14d62ba6ccd77846ba7f1) by RoutineStudio (https://sketchfab.com/TheRoutine) licensed under CC-BY-4.0 (http://creativecommons.org/licenses/by/4.0/)
     - **Closet:** This work is based on "Closet 02" (https://sketchfab.com/3d-models/closet-02-e3072ceec1a94fa29c531f930b637210) by jesusgutierrezc (https://sketchfab.com/jesusgutierrezc) licensed under CC-BY-4.0 (http://creativecommons.org/licenses/by/4.0/)
     - **Laundry Basket:** This work is based on "Laundry basket" (https://sketchfab.com/3d-models/laundry-basket-9e90243fed47475096df9e70c72591

a5) by Karolisbutenas (https://sketchfab.com/Karolisbutenas) licensed under CC-BY-NC-SA-4.0 (http://creativecommons.org/licenses/by-nc-sa/4.0/)
    - **Rug:** This work is based on "Shaggy Rug" (https://sketchfab.com/3d-models/shaggy-rug-64109128f6784c2caaa8cf9904d85043) by GIR__ (https://sketchfab.com/GIR__) licensed under CC-BY-4.0 (http://creativecommons.org/licenses/by/4.0/)
- Sounds from Freesound.org
    - **Correct sound effect:** https://freesound.org/people/Eponn/sounds/421002/
    - **Wrong sound effect:** https://freesound.org/people/TheBuilder15/sounds/415764/
    - **Background music:** https://freesound.org/people/Setuniman/sounds/170373/
- ThreeJS Libraries
    - Pointer Lock Controls: https://threejs.org/docs/#examples/en/controls/PointerLockControls
    - OBJLoader: https://threejs.org/docs/#examples/en/loaders/OBJLoader
    - GLTFLoader: https://threejs.org/docs/#examples/en/loaders/GLTFLoader
    - Loading Manager: https://threejs.org/docs/#api/en/loaders/managers/LoadingManager
    - Vector3: https://threejs.org/docs/#api/en/math/Vector3
    - Vector2: https://threejs.org/docs/#api/en/math/Vector2
    - Box3: https://threejs.org/docs/#api/en/math/Box3
    - AudioLoader: https://threejs.org/docs/#api/en/loaders/AudioLoader
    - Audio: https://threejs.org/docs/#api/en/audio/Audio
    - AudioListener: https://threejs.org/docs/#api/en/audio/AudioListener
- CodePen:
    - Inspiration for “fake” physics and camera control: https://codepen.io/olchyk98/pen/NLBVoW