Sokoban

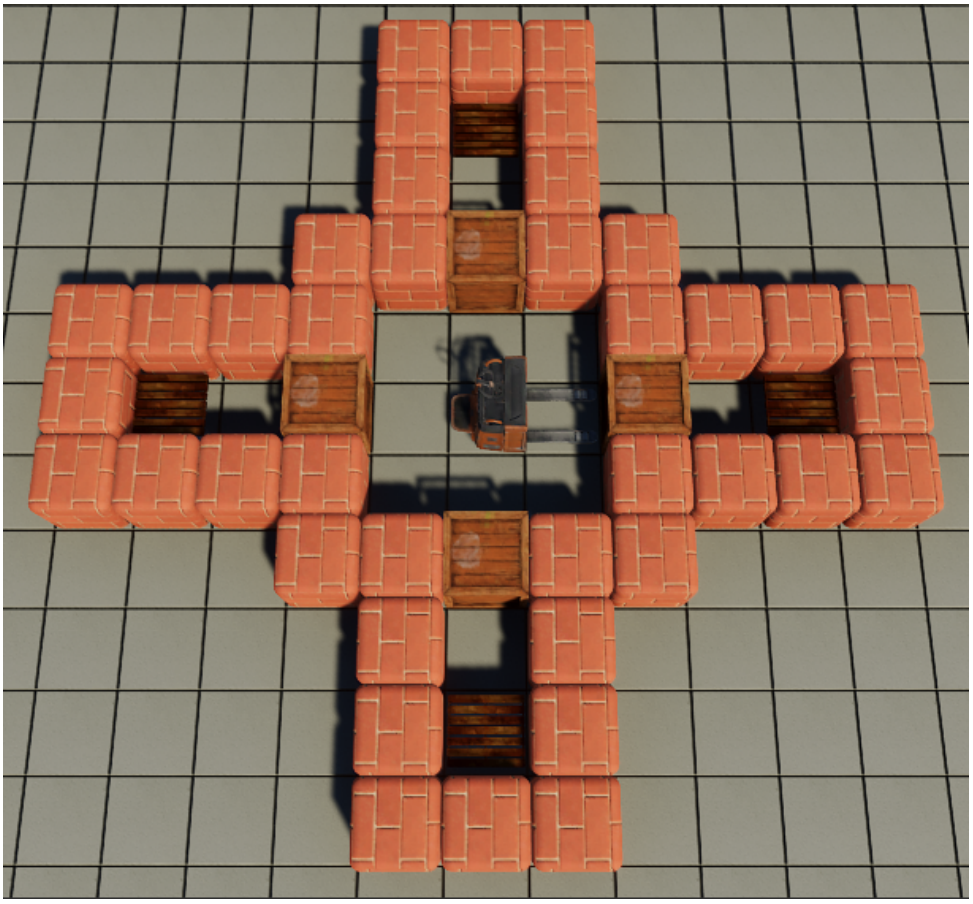
Team 9:
Rodolfo Gomez (25%),
Azeem Mir (25%),
Carlos Rivera (25%),
Alex Najera (25%)

Introduction

In this report, we will be discussing our team's recent project, where we developed a puzzle game using Unity. The report will cover the project description, tools and software used, design and implementation details, possible future improvements, and lessons learned throughout the development process.

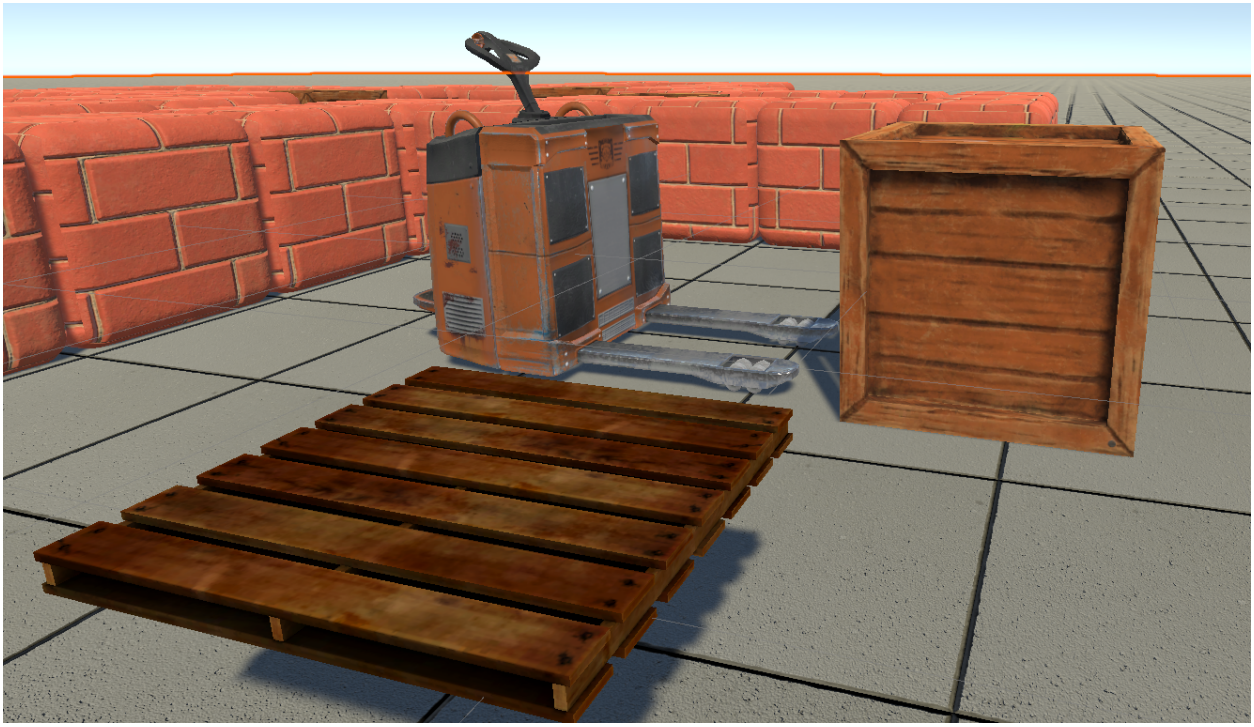
Project Description

Our project is a puzzle game set in a warehouse environment, where the player controls a forklift tasked with pushing wooden crates to specific locations, denoted by wooden pallets. The player can move up, down, left, and right, with turns restricted to a single axis difference in direction. The forklift can only push crates, not pull them. The game starts with intuitive levels that can be solved relatively easily, but as the player progresses, the difficulty level is raised higher and will require strategic thinking and planning to solve. Below is a picture of one of our levels.



Tools and Software Used:

- **Unity**: Our primary game development engine for creating the game world, including the warehouse environment, forklift, crates, and pallets.
- **C#**: We used C# for scripting the game mechanics, such as player movement, crate interaction, and level progression.
- **GitHub**: We utilized GitHub for version control, allowing our team to collaborate effectively and manage different versions of the project.
- **Visual Studio**: Our preferred integrated development environment (IDE) for writing and debugging C# scripts.
- **Unity Asset Store**: We used the unity asset store to source some artifact textures as well as some assets themselves. Below is a picture of some of the assets we obtained.



Design and Implementations Details:

- **Mechanics**:
  - Movements:
    - Game movements generally follow the same movements as traditional 2D Sokoban, with some additional modifications. Player can move up, down, left, and right but also can rotate 90 to orient the forklift in different directions. The player controls the forklift using the keyboard, with the keystrokes mapped as follows: [W,up], [A, left], [S, down], [D, right]. Most PC games use the WASD control scheme, so the movement should be fairly intuitive to players with other PC game experience.
  - Object interaction:
    - Players can only push crates around inside the walled region the distance of a single tile.
    - Players can only push the crates, not pull them. If a crate becomes stuck alongside the wall, the player automatically loses that level and must reset.
  - Camera view:
    - Game employs the use of a top-down camera view to give the player a complete view of the map so strategic moves can be pre-planned using all available information.
- **Design**:
  - Use of assets:
    - Game assets were downloaded from the Unity Asset Store. All assets used are linked in the references section below.
    - Game assets followed the overall theme of the game, which was a warehouse theme.
    - Assets used were a forklift asset, wooden crate asset, wooden pallet asset, concrete tile flooring asset, and a brick wall asset.
  - Level design:
    - In our version of Sokoban, the traditional game of Sokoban was modified such that the player takes up 2x1 units, instead of 1x1. This adds an additional level of complexity to the game, as the player has to consider the orientation of the forklift while maneuvering through the map. Every possible move had to be considered during the conceptual phase of each level design to ensure the player would be able to complete it.
    - Levels were designed using the foundational principles of the traditional Sokoban game, with slight modifications to adjust for the additional rules
    - Each level was designed to increase in difficulty as the game progresses to allow the player to understand the game at first and remain entertained as they advance.
    - Each level was designed in order to ensure that there was a possible path for the player to reach each crate and push the crate into the designated locations
    - The more advanced levels, such as level 4, incorporated certain design strategies in order to increase difficulty and prevent a player who has been able to progress easily from becoming bored.

        - Such design strategies include the use of “traps”, where the seemingly easiest solution would trap the crate, immediately ending the game for the player.
        - Another design strategy was adding easy solutions after the traps, in order to confuse the player. The player is intentionally made to believe the solution is much more complex than it is.

- **Implementation:**
    - Level implementation:
        - Each level was implemented as a single scene. Upon level completion, the scene immediately changes to a scene containing a popup scene.
        - Each level scene consists of game objects to implement the necessary game entities, such as the forklift, crates, walls, and pallets.
    - Objects:
        - Different types of objects have their own scripts and behaviors that allow the puzzle to work as expected. This includes the forklift, crates, targets, and walls.
            - Forklift/Player:
                - The implementation of the forklift was our most complex, needing behavior for its forward movement, recognition of other objects, input, and turning
                    - The forklift was implemented using the ‘Player.cs’ script. The movement behavior was inherited from the ‘Mover.cs’ script, which defines the behavior of the movement for objects in the game.
                    - The script takes player keyboard input and determines the corresponding angle to orient or move the forklift.
                    - The script contains a method to determine whether a turn or movement can take place based on the adjacent game objects.
                    - The forklift can only turn when there is no wall, there is no block next to the base of the forklift, or the block next to the forklift can turn.
            - Crate Object:
                - The crates were implemented using the ‘Mover.cs’ script. Second to the forklift, the crate behavior was the most complex to implement. The following behavior had to be implemented:
                    - Move a single space if the forklift pushes it forward. The animation had to slide it to the next space and set it in place. We used an IEnumerator to do this in a set time, and only allow one push at a time until the forklift activates a push again.
                    - If the space it is trying to be moved to contains a wall block, stay in the current position. This is done

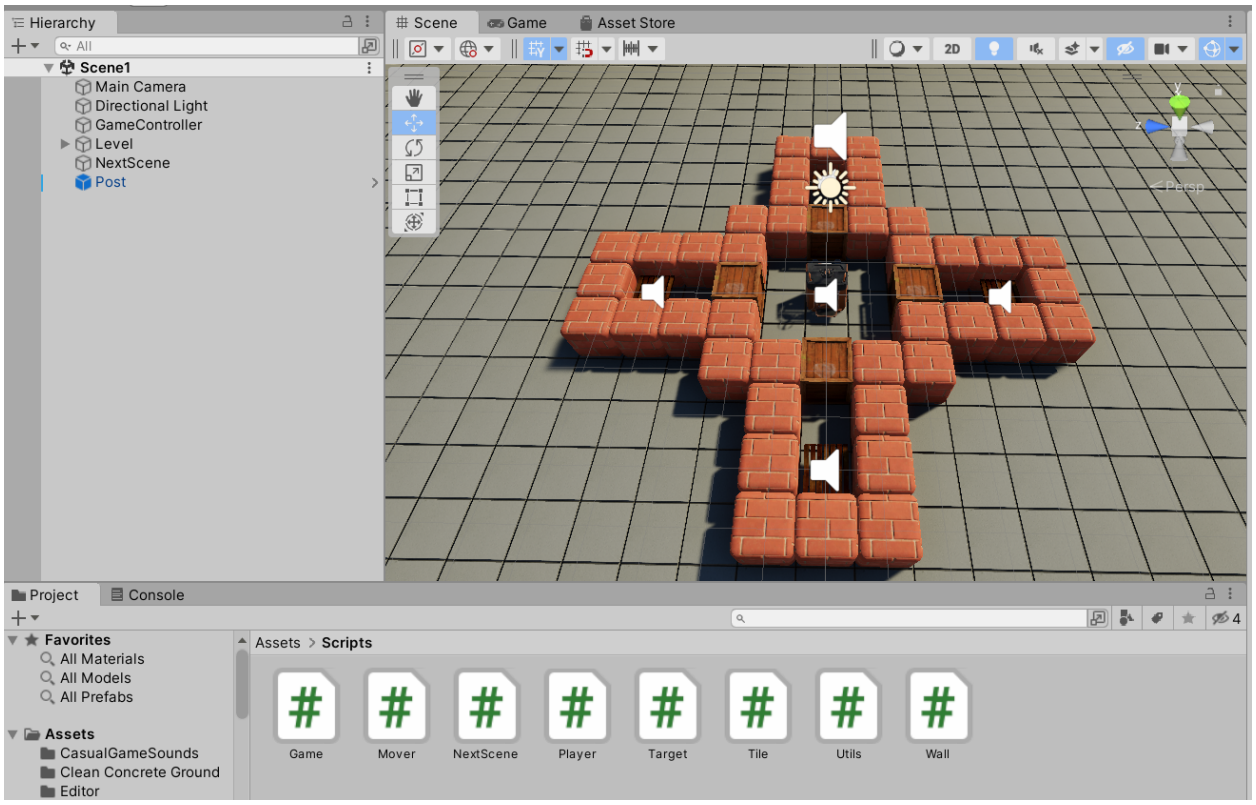
within the Mover.cs script, the can_push_toward function recursively checks if the following space is available, if false, the crate will not move.

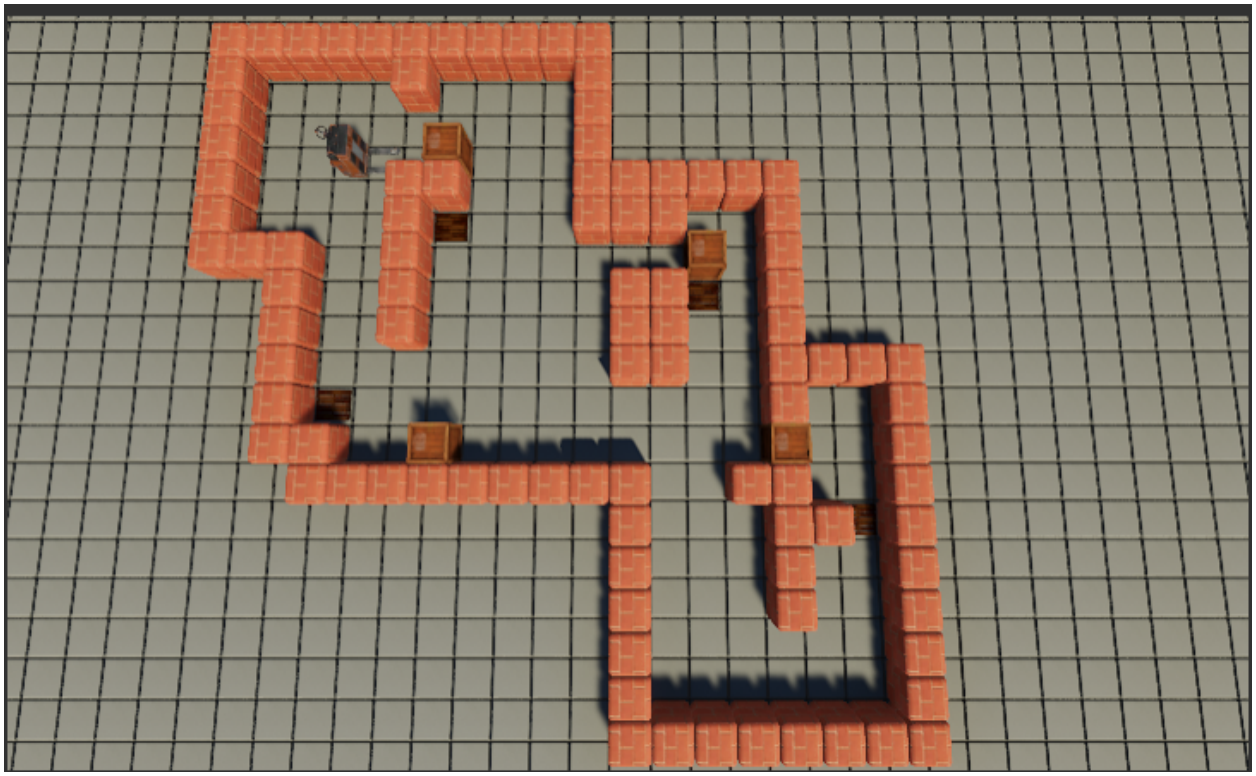
- Target Object:
  - The target is where the pallets are placed, which were acquired from the asset store. Each target and the pallet is a prefab that detects if another object is above it. If the collider is activated, check if that detected object is a crate which then gets flagged as completed.
  - The placement of the targets had to be strategically placed as that was one of the main factors of a level's difficulty.
- Wall Object:
  - Walls were placed as cubes over the individual tiles creating a perimeter around the playable area of each level.
  - The walls are Unity cubes with a brick texture acquired from the Unity asset store.
  - The placement of the walls were our most significant factor of each level. Each being individually placed on the grid allowing for full customization of the level and its size.
- Popup screen:
  - The popup screen upon level completion was implemented by checking if all target locations are filled with crates. When all targets are filled with crates, the current level jumps to the popup screen.
  - The popup screen allows the user to enter the 'N' key to proceed to the next level.
  - Using a counter, the current scene is incremented by one to display the next level scene.

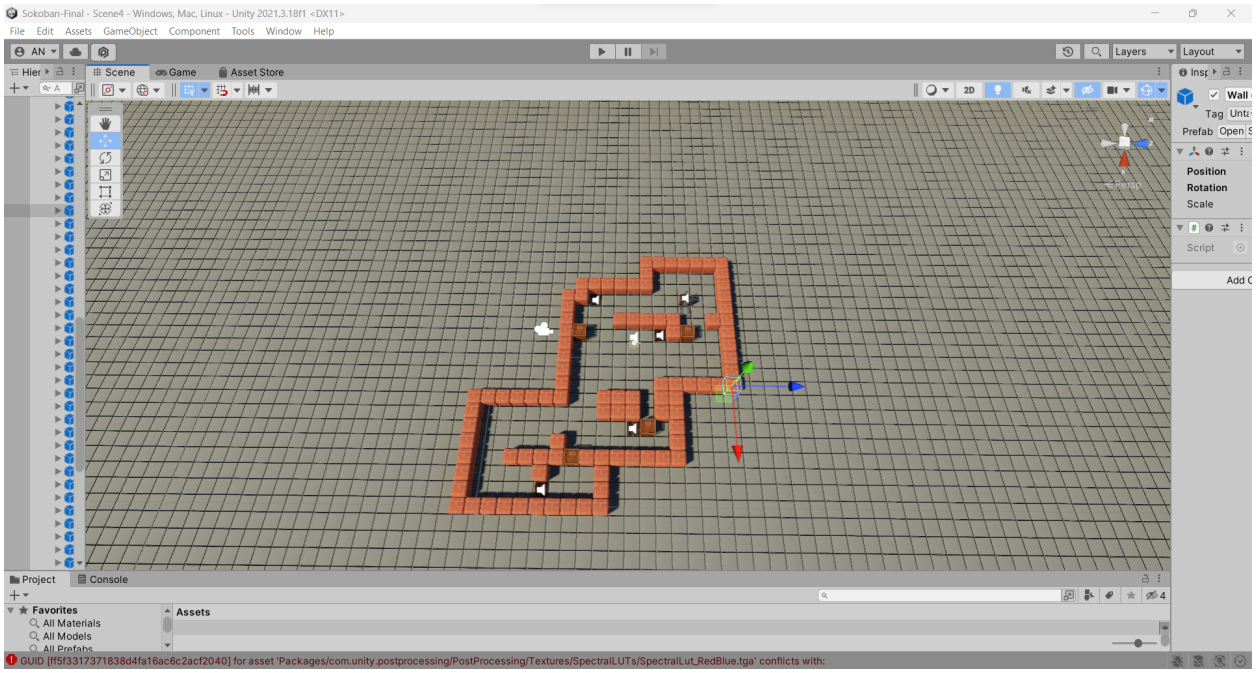
- Sound effects:
  - There were four sounds added to their respective objects. All sounds were acquired from the asset store, and implemented by attaching AudioSources to the most fitting objects:
    - Gamecontroller contains the AudioSource for the level completed sound, with the logic of it added to our Game.cs script. In that script, the condition of all the targets being filled will activate the sound before the Popup scene opens.
    - The Player prefab contains the AudioSource for the forklift engine sound. Within the Player.cs script, the sound activates at the start of every level and continues until the level is complete. Additionally, this prefab also contains the music of the game that also plays constantly while the level is played.

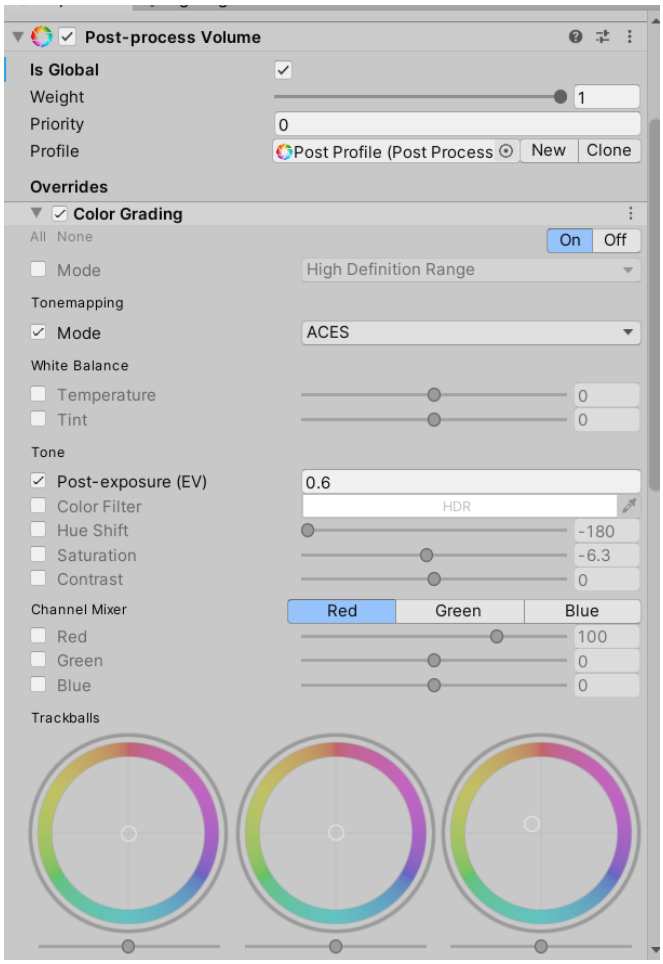
        - The target prefab contains the AudioSource for the target being filled sound. Within the Target.cs script, the sound of a wood scrape will play if a target is filled, audibly signaling that the player has made progress in the level.
    - PostProcessing:
        - We added post processing to our game using the existing post processing asset available in Unity to look less cartoony and more eye-catching. This was added to our MainCamera object and an empty object called Post. The added layers also allowed us to more finely edit the look of the game and the color grading.

Screenshots:






Sokoban-Final - Scene4 - Windows, Mac, Linux - Unity 2021.3.18f1 <DX11>
File Edit Assets GameObject Component Tools Window Help
Layers
Layout
Scene
Game
Asset Store
Position
Rotation
Scale
Script
Project
Console
Favorites
All Materials
All Models
Assets
GUID [ff5f3317371838d4fa16ac6c2acf2040] for asset 'Packages/com.unity.postprocessing/PostProcessing/Textures/SpectralLUTs/SpectralLut_RedBlue.tga' conflicts with:



Lessons Learned:

Throughout the development process, we gained valuable experience and insights, including:

1. **Using Unity to make a game**: We learned how powerful Unity can be graphically and for game development. Working exclusively with it allowed us to learn how to create game objects, scenes, and prefabs, as well as how to utilize Unity's tools to build our levels.
2. **C# coding to make game functions**: We became proficient in scripting game mechanics using C#, including player movement, crate manipulation, and level completion conditions.
3. **Version control using GitHub**: We discovered the importance of version control in collaborative projects and learned how to manage and synchronize our work using GitHub.
4. **Understanding frames in Unity**: We learned about the concept of frames in Unity and how they impact game performance and responsiveness.
5. **Implementing game sounds in Unity**: We learned how to incorporate sound effects and background music into the game to enhance the overall player experience.

6. **Planning and organizing**: We gained valuable experience in project planning, task delegation, and time management to ensure smooth development and timely completion of our project.
7. **Team communication**: We improved our communication skills, allowing for effective collaboration and efficient problem-solving within our team.

Possible Future Improvements:

- Additional game features:
  - Allowing forklift to pick up boxes
  - Adding more sound effects and curating a soundtrack and allowing the user to select songs
  - Adding a timer and point system
  - Adding additional levels varying in difficulty
  - Allowing the player to unlock special modifications as they progress
  - Adding a leaderboard for top scoring players
  - Adding a multiplayer mode to allow players to complete each level collaboratively
  - Adding functionality for a player to come up with their own level designs
  - Adding functionality for the user to zoom and pan the camera in/out during gameplay
- Aesthetic improvements:
  - Adding additional background scenery to each level in order to give the game a more realistic feel
  - Adding variety to each level by incorporating different settings and unique designs for each level
  - Fine tuning the post processing and improving the lighting

Conclusion

In conclusion, our team developed a captivating and challenging warehouse puzzle game using Unity, C#, GitHub, and Visual Studio. The project allowed us to explore various aspects of game development and computer graphics, from creating game objects and scenes to implementing game mechanics and sound effects. Moreover, the Unity Asset Store proved to be a valuable resource for enhancing the visual appeal of our game with high-quality assets and textures. Throughout the development process, we gained hands-on experience in project management, task delegation, and time management, ensuring the smooth progression and timely completion of our project. We learned the significance of efficient team communication in collaborative projects, which enabled us to tackle challenges and devise creative solutions more effectively. As we reflect on our project, it is evident that the skills and insights we have gained in game development, programming, version control, and teamwork will undoubtedly prove beneficial towards our future careers. This project has not only given us a solid foundation in game design but has also fostered a sense of camaraderie and a stronger appreciation for the collective efforts of our team. As we move forward, we will continue to build upon these lessons and experiences, striving to create even more engaging and innovative games in the future.

References:

- Wooden Pallet Pack. Pixel Games, 2014. Unity Asset Store, https://assetstore.unity.com/packages/3d/props/industrial/wooden-pallet-pack-657.
- Forklift. Nova Shade, 2014. Unity Asset Store, https://assetstore.unity.com/packages/3d/props/industrial/forklift-60960.
- World Materials Free. Avionx, 2019. Unity Asset Store, https://assetstore.unity.com/packages/2d/textures-materials/world-materials-free-150182.
- Clean Concrete Texture. Pixel Indie, 2012. Unity Asset Store, https://assetstore.unity.com/packages/2d/textures-materials/concrete/clean-concrete-texture-37028.
- Worn Wooden Crate. Hocker, 2021. Unity Asset Store, https://assetstore.unity.com/packages/3d/props/furniture/worn-wooden-crate-246735.