COSC3072/COSC3073 | Games Studio 1

Assessment 3: Implementing a 3D Clone of Marble Madness

Students:
Tran Phan Hoang Phuc – s3929597
Nguyen Doan Trung Truc – s3974820
Nguyen Vo Truong Toan – s3979056

Tutorial Session: Thursday 12:30 PM – Group 4 SGS

Lecturers: Dr. Nhat Quang Tran & Dr. Kapil Dev

Due Date: September 21th – 23:59

"We declare that in submitting all work for this assessment we have read, understood and agree to the content and expectations of the Assessment declaration (Links to an external site.)"

I. Game Design and Mechanics

3D Marble Madness is a physics-based puzzle platformer where players control a marble ball through 3D levels. The core objective is to navigate from the starting position to the red target zone before the timer expires.

1.1) Player Character - The Marble Ball -

The player character is implemented as a **RigidBody3D** with the following specifications:

- **Mass**: 100.0 kg
- **Rolling Force**: 40.0 (controls acceleration)
- **Jump Impulse**: 100.0 (vertical jump strength)
- **Gravity Scale**: 1.0 (standard physics gravity)

1.2) Movement System -

Movement is achieved through **angular velocity** manipulation rather than direct translation, creating authentic rolling physics based on the code structure:

```
if Input.is_action_pressed("forward"):
    angular_velocity.x -= rolling_force * delta
elif Input.is_action_pressed("back"):
    angular_velocity.x += rolling_force * delta
if Input.is_action_pressed("left"):
    angular_velocity.z += rolling_force * delta
elif Input.is_action_pressed("right"):
    angular_velocity.z -= rolling_force * delta
```

1.3) Respawn System

The game implements a checkpoint-based respawn system that saves players if they fall off the track by having a safety Area3D underneath the entire game level. This ensures player returns to the exact position they last were right before death with reset velocity

1.4) Enemy AI System

Enemies inside the game use CharacterBody3D node base with custom behaviour featuring 3 different states. First, the enemy stands still until it detects the player ball; once it detects the player in range, it will track and follow the player ball. Finally, it will attempt to knock them off/push them back off the platforms on contacts. It’s implemented using physics impulses for realistic collisions and interactions.

1.5) Moving Platforms

Moving platforms use **AnimatableBody3D** nodes with **Tween** animations, allowing players to ride them naturally through Godot's built-in kinematic body interactions.

II. Physics System Implementation

2.1) Godot Physics Integration - The game leverages Godot 4's built-in physics engine with custom **PhysicsMaterial** resources applied to the ball. The physics material controls friction (0.3-0.7) and bounce (0.2-0.4) properties, fine-tuned to create satisfying marble-like behavior that balances controllability with realistic physics simulation.

2.2) Camera System - A detached camera system follows the ball smoothly using top-level positioning:

```
camera.top_level = true
camera.global_transform.origin = global_transform.origin
```

2.3) Moving Platform Physics - Moving platforms use relative positioning for easy placement:

```
var end_position = start_position + move_distance
tween.tween_property(self, "global_position", end_position, move_duration)
```

III. Tileable 3D World Structure

3.1) GridMap Architecture

The level design utilizes Godot's **GridMap** system with a custom **MeshLibrary** containing six distinct block types:

1. **Floor** - Basic platform tiles
2. **Floor-border** - Edge pieces with raised borders
3. **Floor-corner** - Corner transition pieces
4. **Floor-corner-borders** - Corner pieces with protective edges
5. **Floor-ramp** - Sloped transition pieces
6. **Target** - Goal/victory zone markers

3.2) Grid System Specifications

- **Cell Size**: 4.0 x 4.0 x 4.0 units
- **Octant Size**: 8 (subdivision factor for optimized rendering)
- **Physics Integration**: Automatic collision generation for all placed blocks

This modular approach allows rapid level creation while maintaining visual consistency. Each block type serves specific gameplay purposes, from basic navigation (floor) to safety features (border pieces) and objective markers (target).

IV. Challenges and Solutions

Challenge	Issue	Solution
Moving Platform Death Loops	Players falling from moving platforms would respawn at positions where the platform had moved away, creating infinite death cycles	Implementation of safety zones using Area3D triggers positioned beneath moving platforms that override the standard respawn system
Enemy Model Integration	Replacing simple geometric enemies with complex 3D models while maintaining physics compatibility and AI behavior	Separated visual representation from collision detection by using the imported Godot mascot model as a child of the enemy's collision system
Physics Tuning Balance	Balancing realistic physics with responsive gameplay controls, preventing the ball from becoming too sensitive or too sluggish	Extensive playtesting and iteration of rolling force values, physics materials, and mass distribution to achieve responsive control while maintaining physical believability
GridMap Rotation Issues	Inability to rotate entire GridMap layouts after construction, requiring manual rebuilding when orientation needed adjustment	Developed workflow to plan level orientation from start and use individual block positioning rather than post-construction rotation

V. Individual Contribution

5.1) Nguyen Doan Trung Truc - s3974820

Tasks I was responsible for:

- Level 0 (tutorial level), Level 1, Level 2 and Level 3 with appropriate difficulty scaling for new players
- Implement user interface systems including main menu, pause menu, and game over screens
- Design and code the scoreboard and timer functionality with proper countdown mechanics
- Implement enemy AI systems including pathfinding, chase behavior, and collision detection
- Create victory zone detection and level transition logic
- Develop death zone mechanics and respawn trigger systems
- Implement sound effects integration and audio management

- Main game testing, debugging, and performance optimization

Challenges faced and how I overcame them: The biggest challenge was implementing the enemy AI system since this was my first time working with NavigationAgent3D and complex state management in Godot. Creating smooth enemy movement that felt natural while maintaining good performance required extensive testing and iteration. Another major hurdle was integrating the UI systems with the game state management - ensuring the timer, score, and menu transitions worked seamlessly across different levels without conflicts. Git collaboration also presented challenges when working on interconnected systems like UI and gameplay logic simultaneously with teammates. I overcame these by studying Godot documentation extensively, creating modular code structures for the AI states, and establishing clear communication with my team about shared systems. Regular testing sessions helped identify and fix integration issues early.

Estimated contribution: I estimate my contribution at approximately 33% of the total work, consistent with the balanced distribution among all team members.

Team dynamics: Working with this team was highly collaborative and productive. Everyone had clear responsibilities but was always willing to help debug issues or brainstorm solutions. The combination of different technical strengths made problem-solving more effective and kept the development process engaging.

5.2) Nguyen Vo Truong Toan - s3979056

Tasks I was responsible for:

- Drafting Tutorial level, and the three main levels.
- Creating environmental lighting and sky color for each level.
- Adding sound effects and background music for each level.
- Designing level progression flow and difficulty across three main levels.
- Creating timer system integration with level-specific time limits.
- Testing all levels and refining gameplay balance.
- Adding UI buttons and HUD elements.

Challenges faced and how I overcame them: Level design proved more complex than anticipated, especially balancing difficulty progression while maintaining engaging gameplay flow. Each level needed to introduce new challenges without overwhelming players, requiring multiple iterations to achieve the right pacing. Working with the GridMap system had limitations - particularly when trying to create more organic, curved paths or complex vertical arrangements. The constraint of using modular blocks meant being creative with spatial relationships and obstacle placement. Coordinating lighting and music with the overall game mood while ensuring performance remained stable was also challenging. I addressed

these by playtesting extensively with teammates, studying level design principles from similar games, and iterating on lighting settings to achieve the desired atmosphere without impacting frame rates.

Estimated contribution: I believe I contributed approximately 33% of the total project, equal to my teammates' contributions.

Team dynamics: I had a great time working on this project with my teammates. We maintained excellent communication throughout development. Regular feedback sessions on level design helped refine the gameplay experience, and everyone contributed creative ideas that enhanced the three main levels. The collaborative environment made the iterative design process much more effective.

5.3) Tran Phan Hoang Phuc - s3929597

Tasks I was responsible for:

- Design and implement Level 0 (tutorial level), Level 1, Level 2 and Level 3 with appropriate difficulty scaling for new players
- Develop moving platform systems using AnimatableBody3D and Tween animations
- Create safety net mechanisms to prevent infinite death loops on moving platforms
- Implement platform movement patterns including horizontal, vertical, and timed sequences
- Implement weaker type enemy AI systems including pathfinding, chase behavior, and collision detection
- Code relative positioning systems for easy platform placement across levels
- Design tutorial-specific gameplay elements and player guidance systems
- Test and refine platform physics interactions with the marble ball
- Implement Area3D trigger systems for respawn override functionality

Challenges faced and how I overcame them: The most significant challenge was solving the moving platform death loop problem where players would fall and respawn only to find the platform had moved away, creating infinite death cycles. This required developing a completely new safety net system using Area3D triggers that override the standard respawn mechanics. Another major challenge was creating moving platforms that felt natural to ride while maintaining the physics-based marble movement - balancing the platform movement speed and patterns so players could realistically time their jumps and navigation.
Implementing the tutorial level required careful consideration of player psychology and learning curves, ensuring new players could understand the game mechanics without explicit instruction. I overcame these by iterating on the safety zone placement and trigger sizing,

extensive playtesting with different movement patterns, and studying other games' tutorial design approaches to create an intuitive learning experience.

Estimated contribution: I estimate my contribution at approximately 33% of the total work, consistent with the balanced distribution of responsibilities among all team members.

Team dynamics: The collaborative environment was excellent for tackling complex technical problems. When the moving platform death loop issue arose, the entire team contributed ideas and testing, which led to the safety net solution. Regular communication helped ensure my tutorial level design aligned with the difficulty progression of the main levels created by my teammate.