**WORKING TITLE: Slice Don’t Slash**

**Overview**

In Slice Don’t Slash, a third person action physics game; the player takes control of a unnamed warrior. Woken up in a battlefield surrounded by monsters trying to kill anything in their path. You must make it to the end. Fight at all costs using your gifts on the battlefield and strike your way to victory.

**Gameplay Mechanics**

Marko will be focusing on completing the player Guard mechanic. This will be using a state where we can block incoming attacks from an enemy. This will add a small knockback for 0.5 seconds which will cause the enemy to reset and recalibrate to attack again. The second physics mechanic which will be used is the knockback. Which will use code from the push assignment we used in Physics with Sergio. This piece will be very versatile for the physics we are using as the movement must be implemented using Kinematics.

For AI, Marko will also add an enemy guard so that it blocks the player at specific points when attacking. Using the player guard state, it should be possible to implement that script with a timer depending on the enemy health since that is how we are using the states for the enemies. The second AI function implemented by Marko will be the boss mechanic where they will be striking for the player and when missing, they will get stuck in the ground and have a few seconds to get out of it. This will be the only time in which the player can damage the enemy.

Ross will add an enemy roll implementation where the enemy will attempt to avoid attacks using roll when in a specific health pool. Using states, the player code should be used to efficiently give both characters maximized abilities. Ross will then add parry to the guard mechanic. This will give the player a small timeframe in which we can use similarities from knockback to stun the enemy to open them up for an attack.

For AI, there will also be an addition for when an enemy is almost dead. This will be during its FreakingOut state, where it will throw a trap on the terrain and that trap will explode on proximity with the player.

We will attempt to add in a slice mechanic as well once we have finished tasks. This will be the cherry on top where you can slice through enemy meshes once they are FreakingOut.

**AI Behaviors**

The enemy AI in Slice Don’t Slash will consist of three enemies. two smaller monsters and one boss-like enemy. The starting enemies will have six states; Idle, Patrol, Attack, Damaged, AlmostDead and FreakingOut. These states will all include various mechanics. The Idle state will be upon Entry. Leading the enemy to patrol through a designated area to eventually find the player. Leading it to an Attack state, the will cause the enemy to attack the player on a cooldown. The Damaged state will occur once the player has hit the enemy two times. At which point, the enemy will attack at a much smaller cooldown. Causing a flurry of attack in which the player must dodge, or guard correctly. This enemy will also have a cooldown in between flurries so that the player can counter.

Once the enemy has been damaged enough to be in AlmostDead state, the enemy will be in guard much more frequently than before. Attempting to switch it’s tactic with the player. This will cause the player to pick their spots, and choose precision rather than a constant attack and drain their stamina. The final enemy state will be FreakingOut, in which the enemy will toss traps in attempt to destroy the player, even after death. These mines will be planted wherever they land, and will not activate until the player is in proximity to them.

The boss will have three states. Attacking, Blocking and Stuck. Blocking will be its natural state to make the boss unkillable unless it’s Stuck. Attacking will cause AOE damage so that the player has to move out of the way and then return, since when the boss’s attack misses, it becomes Stuck and that will be the only opportunity in which the boss can be damaged.

The Slice mechanic, if we planned this well enough, will be used when an enemy is close to death and can be used as an execution to slice the enemy mesh into pieces and finish them off.

**Physics**

The physics for this game will control the player. Hardcoding all aspects of player controls except for the camera. States will include Idle, Movement, Attack, Roll, Guard, Parry, Knockback. The Idle state will be upon entry of the game, bringing the Movement state to occur based off of input. This moves the player through the map and will be the most used State. Attack will occur from an input which will cause the player to attack in a specific direction in which the player is looking. This is to damage the enemy. Roll will be the primary and only dodge for the player to escape the line of fire without blocking or parrying. This will cause an avoidance to explosions or AOE effects. Guard is going to be used to block primary attack from the enemy. This will disrupt the enemy for 0.5 seconds and cause them to reset so they may attempt to attack again or give the player an opportunity to counter. Parry will be a short time frame in which the player has allocated to cause a stun/knockback to the enemy for two seconds. This will give the player a better opportunity than blocking but will give the player the player a more difficult challenge if they were to succeed. The Knockback will occur when a combo is landed, multiple attack in succession will be considered a combo.

The Slice mechanic, if we planned this well enough, can be an additional mechanic where the player can use their remaining stamina to kill an enemy by slicing their mesh using a plane and finish them off.

**Production timeline:**

***Week 1:***

**(\*Check with Sergio if everything needs to hardcoded or just assigned tasks\*)**

**\*Bonus\* Finishing Mechanics if possible.**

Slice Mechanic

Using physics in Unity, to slice meshes along a plane to hack off limbs and kill enemies around you.

*Time Estimate: 5 Hours*

*Ross/Marko*

**Task 1:**

Create working GitHub for both of us to share the project efficiently, and work through errors.

*1 hour*

*Marko/Ross*

**Task 2:**

State separation planned and added to player in order to activate different states based off of input. Attacking, Dodging, Guard, Parry, Knockback.

private enum PlayerStates {Idle, Movement, Attack, Roll, Guard, Parry, Knockback};

PlayerStates currentState;

currentState = PlayerStates::Idle;

switch (currentState)

Each state will require Kinematic code, which can be broken down to week two in order to fully implement core functionality of the game.

*Time Estimate: 2 hours*

*Ross/Marko*

**Task 3:**

Basic Movement Rigidbody IsKinematic and nonKinematic for Prototype  
 Being able to move the character using Unity helpers is step 1 to get an idea for game. Step two is to implement an IsKinematic hardcoded version to fulfill project guidelines.

*Time Estimate: 2 Hours*

*Marko/Ross*

**Task 4: Physics Task #1**

Basic roll implementation added. Using an isKinematic using physics to push the player in the direction it’s facing. This can tuned Week 2 to roll in the direction of the mouse/left analog on gamepad.

*Time Estimate: 5 hours*

*Ross*

**Task 5: Physics Task #1**

Guard implementation added to block attacks from enemies. This will be using a game object and a collider to block when activated incoming attacks. Not damaging the player.

*Time Estimate: 5 hours*

*Marko*

**Task 6:**

State separation planned and added to enemy in order to activate different states based off of enemy health. Attacking, dodging.

Ex.  
private enum EnemyStates {Idle, Patrol, Attack, Damaged, AlmostDead, FreakingOut};

EnemyStates currentState;

currentState = EnemyStates::Idle;

switch (currentState)

Each state will require Kinematic code, which can be broken down to week two in order to fully implement core functionality of the game.

*Time Estimate: 2 hours*

*Ross/Marko*

**Task 7: AI Task #1**

Core basics of enemy attacking more when in Damaged state which will make the enemy attack more often. Using health to trigger the specific damaged state.

*Time Estimate: 5 Hours*

*Ross*

**Task 8: AI Task #1**

Core basics of enemy guard when the player is attacking implemented. This will occur when the enemy is AlmostDead. Using health to trigger the specific damaged state.

*Time Estimate: 5 Hours*

*Marko*

***Week 2:***

**Task 1: Physics #2**

Implement knockback when hitting an enemy with a player. IsKinematic to push the enemy back when they take a hit. This will cause a 1.5 second delay between the enemy movement and attacking.

*Time Estimate: 4 Hours*

*Marko*

**Task 2: Physics #2**

Add a parry to add onto the guard, using specific timing when guard is activated to match when the strike connects in order to parry the attack from the enemy causing a 2 second delay between enemy movement and attacking.

*Time Estimate: 4 Hours*

*Ross*

**Task 3: AI Task #2**

Adding an enemy trap that’s thrown when the enemy is FreakingOut, this would be an obstacle placed by the enemy that would activate when in proximity of the player causing damage.

*Time Estimate: 4 Hours*

*Ross*

**Task 4: AI Task #2**

Enemy boss strikes and misses and causes its weapon to be stuck in the ground. This will be the only opportunity to strike the boss as it will be guarding for the whole instance.

*Time Estimate: 4 Hours*

*Marko*

**Task 5: AI**

Clean up. This will be allocated time for both Ross and Marko to adjust the boss states and adjust the timing for when enemies attack. Setting specific timings for when enemies are attacking so they are not overlapping, and that the boss shows up once two enemies are killed.

*Time Estimate: 4 Hours*

*Marko/Ross*

***Week 3:***

**Task 1: Clean Up**

Kinematic scripts for player. Make sure that the physics for the player is adjusted accordingly and that the player inputs, controls are all functioning properly regarding controller use. Adjust accordingly.

*Time Estimate: 5 Hours*

*Marko*

**Task 2: Clean Up**

Kinematic scripts for enemy. Make sure that the physics for the player is adjusted accordingly and that the enemy is functioning as it is supposed to. Following it’s states and that the balance in mechanics is correctly being used.

*Time Estimate: 5 Hours*

*Ross*

**Task 3: Bonus Ability, #SLICE**

Adding a slice mechanic to slice enemy meshes when they are almost dead. This is a bonus task if we have time to finish it and add it into the game. This will be used with a plane slicing through a mesh and we could add physics to move the sliced object so that it moves away from its previously sliced mesh.

*Time Estimate: 6 Hours*

*Ross/Marko*

**Task 4: Final Push**

Add in remaining required code, or fixes to polish the game. As this will most likely occur throughout the project. Allocating specific time at the end would be most beneficial in order to come out with a final completed and clean version of the game demo that is portfolio ready.

This could include, code, animations, assets and various UI in order to make the most out of the game.

*15 Hours*

*Marko/Ross*