**WORKING TITLE: Slice Don’t Slash**

**Overview**

In The Faithful, 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.

**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:**

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 3: 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 4: 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 5:**

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 6: 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 7: 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*

**Gameplay Mechanics**

REPLACE THIS TEXT WITH CONTENT: Describe exactly which 2 physics and 2 AI features (per person) from our course will be part of your project. This is basically a contract that indicates what you will be graded for, it’s a commitment. Make sure it presents a good level of challenge, it’s not about simply grabbing the implementations that are already done and throw them together. Make sure it involves some work.

**AI Behaviors**

REPLACE THIS TEXT WITH CONTENT: Be very clear on which feature in your game uses which AI algorithms. Make sure its clear exactly which states or behaviors will be part of which enemy.

**Physics**

REPLACE THIS TEXT WITH CONTENT: Same as previous subsection, but for physics. Make sure you implement the physics logic yourself (Rigidbody with IsKinematic enabled). You can use the built in Unity physics engine for additional complementing features if you want to flesh out more your concept, but at least 2 features should present the challenge of being done manually (similar to the assignments that have been delivered).

Sergio

Roll to escape danger

Based on direction

Knockback on hit

// Bonus Slice equation then use kinematic after slice

Rely on ragdoll physics of unity

Guard with parry detect orientation of attack – range of parrying with shield

SPECIFIC TUTORIALS for Sections

AI

=Health based attacks

=Beginning not dodging

=State separation using health of enemy

=Trap set within level that activate based on position of player

Temporary helper (distraction)

MAYBE BOSS FIGHT?

=Weapon stuck – perfect enemy |