Dimfrost Studio

Programming Work Test

This work test will be centered around the Advanced Locomotion System (ALS) for Unreal Engine. This is a complex system for character movement and animation that we use in Bramble: the Mountain King.

You can find ALS for free on the Unreal marketplace:
https://www.unrealengine.com/marketplace/en-US/product/advanced-locomotion-system-v1
Please install it and open the demo project.

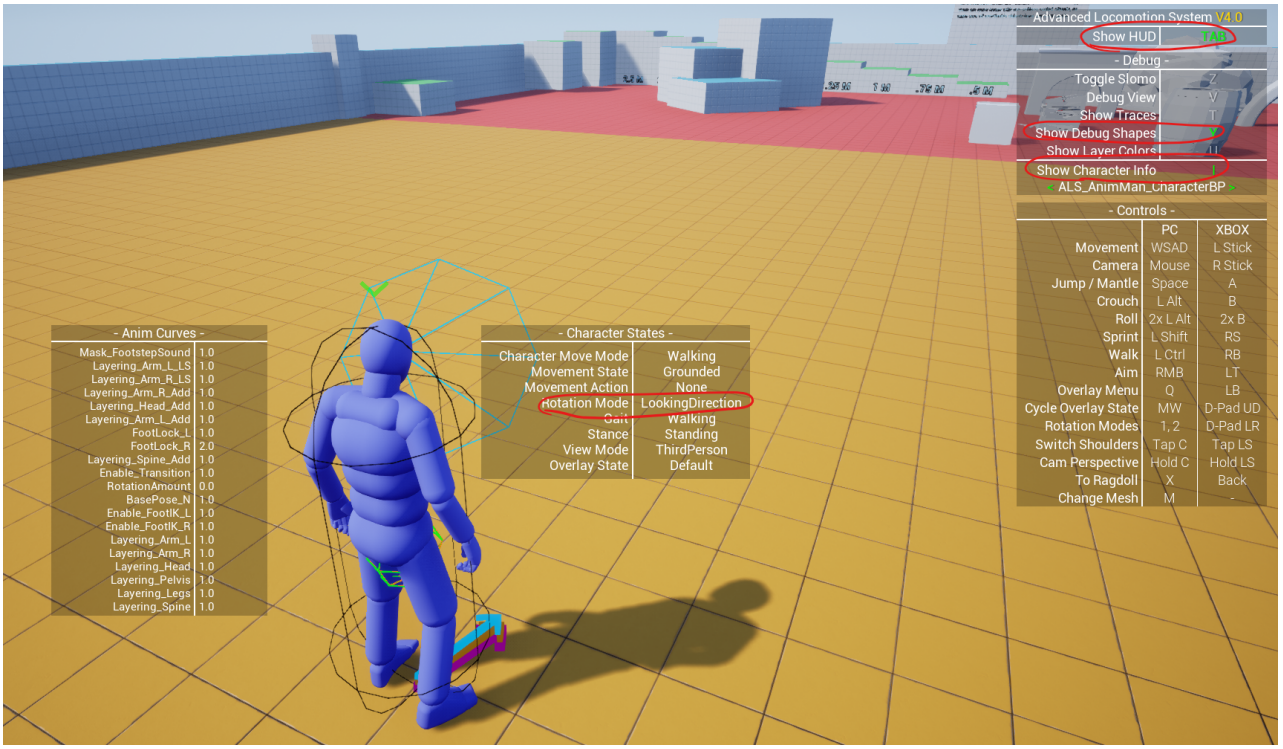
The tasks are of increasing complexity. The first task doesn't involve reading or writing any code while the last task involves analyzing a complex system and can be very difficult. Please complete the tasks to the best of your ability. In case you cannot complete a task, please elaborate on which parts are the most challenging and what information you would require to complete it.

This work test includes answering questions in writing and making modifications to the ALS demo project. Please present your work in the form of a document with answers written in **English** and provide a link to your modified project.

Task 1: Character Movement

If you play the ALS demo project, you can run around in a test scene. The playable character has various movement features. You can start by trying to move around in the scene and testing the different functionality of the character.

The project supports printing debug info about the character to the screen. While doing this task, you will need to press **Y** to show debug shapes and press **I** to show character info. In the "Character States" info box, you can see a variable called "Rotation Mode". The character has three different rotation modes: "VelocityDirection", "LookingDirection" and "Aiming". You can switch between the first two by pressing the **1** and **2** buttons on the keyboard. You can enter the "aiming" mode by holding the right mouse button.

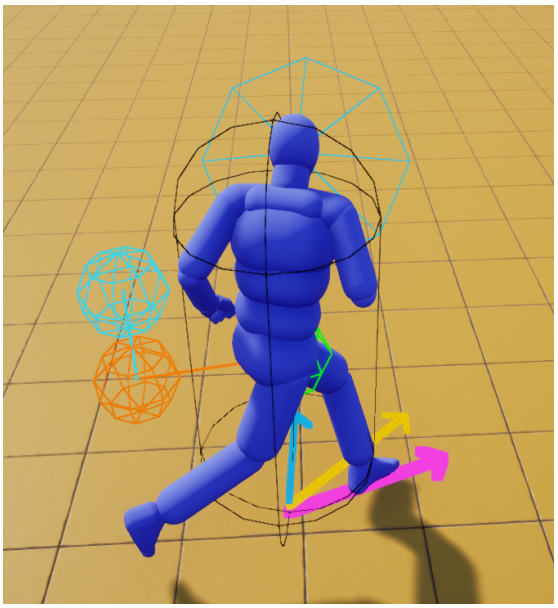


Task 1.a

- After you've tested switching between the different rotation modes, describe how each one affects the character. Explain the relationship between **Input Direction**, **Character Rotation** and **Camera Rotation** for each mode. Try to cover all situations, such as rotating the camera while the character is standing still. You *do not* need to look at any code for this task.

Task 1.b

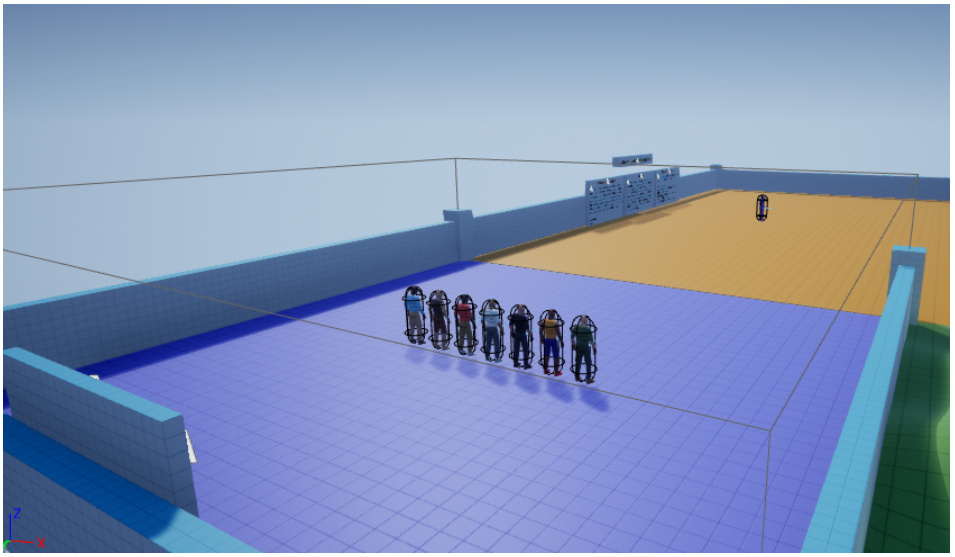
- Look at the three arrows drawn below the character when debug shapes are shown. What does each arrow represent? You *do not* need to look at any code for this task.



Task 2: Red light, green light (Squid game)

You are to implement the “red light, green light” game as portrayed in the show “Squid Game” inside the ALS demo scene. If you are not familiar with the rules, an explanation can be found here: https://en.wikipedia.org/wiki/Statues_(game). Making minor modifications to the rules is ok.

The ALS demo scene contains a blue area with different AI characters walking around. Please modify the behaviour of these characters to participate in the game along with the players. You should use the large space there as the game area, and add the finish line and guardian actor at the other end.



Required features:

- An actor that periodically turns around to look in the direction of the participants. If any participant is moving while the actor is looking back, that participant should be eliminated (tip: call the “RagdollStart” function of the ALS character). There should be some random variation in how long the actor stays in one state, and how long it takes to turn.
- Use the AI actors already present in the scene as participants in the game along with the player character. The AI should run forward towards the finish line and stop approximately when the actor turns back each time. Implement this by modifying the behaviour tree for the AI. Also add some variance to their movement so that some of them will fail and get eliminated.
- If a participant passes the finish line, they should be safe and unable to be eliminated after that point.
- There should be a timer which counts down the game time. When the time is up, all participants that haven’t passed the finish line should be eliminated. The timer needs to be displayed in some way (e.g. a print string or UI text).

Completely optional features (just for fun):

- Add a chanting sound effect to the actor, as in the show.
- Make sure that the player's movement won't be detected if standing behind an AI participant.
- Add visual effects for the players being eliminated.

Make your implementation in blueprints, and try to write well-structured code. Please also briefly summarize your implementation and where to find your modifications.

Task 3: Mantle system (advanced)

ALS includes a system for climbing ledges, or "mantling". If you walk up to a wall and press the jump button, the character will climb the wall provided there is space to stand on top. For this task, you will analyze and make small changes to this system.

First, play around in the demo project to get an intuitive feel for how the mantling works. The character can both climb ledges while standing and catch ledges while in the air. There are also separate animations depending on the height of the ledge. For this task, you will investigate the functions "MantleCheck", "MantleStart" in "ALS_Base_CharacterBP". You might also need to look up other functions to get context for how certain variables are created or used. It might be helpful to press **T** to enable traces to be drawn.

Task 3.a

- Explain the implementation of the "MantleCheck" function. How does the system determine whether or not the character should be able to mantle, and how is the end position calculated? What parameters affect the ability of the character to climb a ledge? How are the different mantle types selected? It might be helpful to include drawings to illustrate your explanation.

Task 3.b

- Describe the "MantleParams" variable: how are its member variables generated and what is it used for? Please include explanations about the individual members of the struct.

Task 3.c

- Please explain the purpose of the "position / correction curve". How are the x, y and z components used? What is the relationship between these curves, the "starting offset" and the mantle animation montage? Hint: find the mantle animation montages from the content browser and take a closer look at the animation.

Task 3.d

- Consider the situation of a programmer and an animator wanting to implement a new mantle animation. Would you think the ALS mantle system is easy to work with? Why or why not?

Task 3.e (programming)

- If the player stands in front of a ledge and pushes the jump button without moving forward (i.e. no movement input), the character does not mantle. Why? Make modifications to the code such that the character will still mantle in this scenario. Describe the changes you made to achieve this result.

Task 3.f (programming)

- Go inside the demo scene and find the structure with three blocks from the picture below. Modify the mantling to make sure that if you **jump** from the ground to the tallest block, you will be able to catch and climb up the ledge **in air**. Modify the relevant parameters to make the animation look smooth. The arms should not flap up and down in the mantle motion. Do this only by changing parameters in structs. You do not have to change any animation itself. Make sure to describe your implementation.

  Hint: check the overrides in ALS_AnimMan_CharacterBP
  Caution: this task concerns the in-air mantle and not the grounded mantle