Technical Documentation

ProjectBashBall

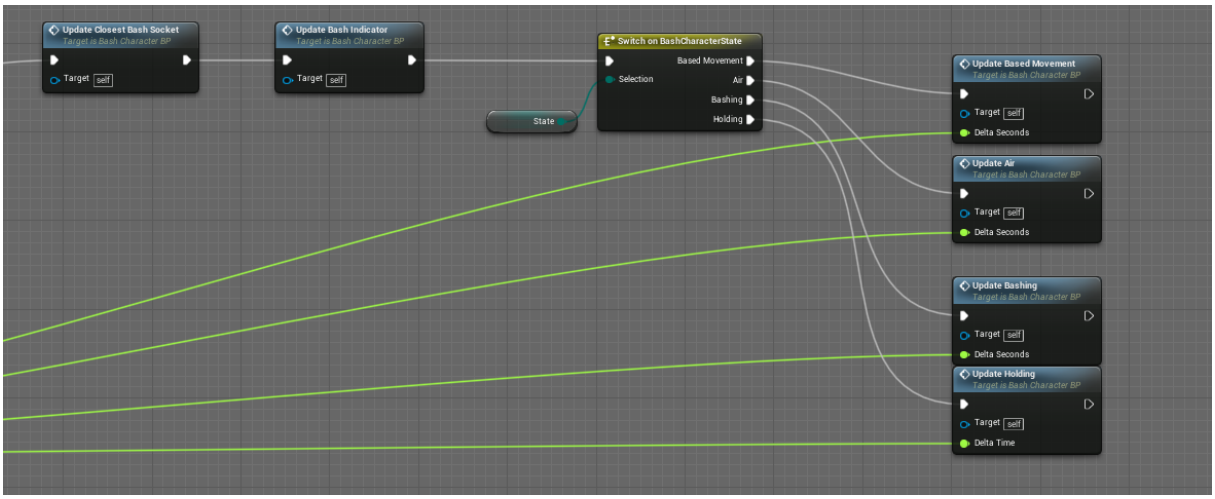
Ms495

Overview

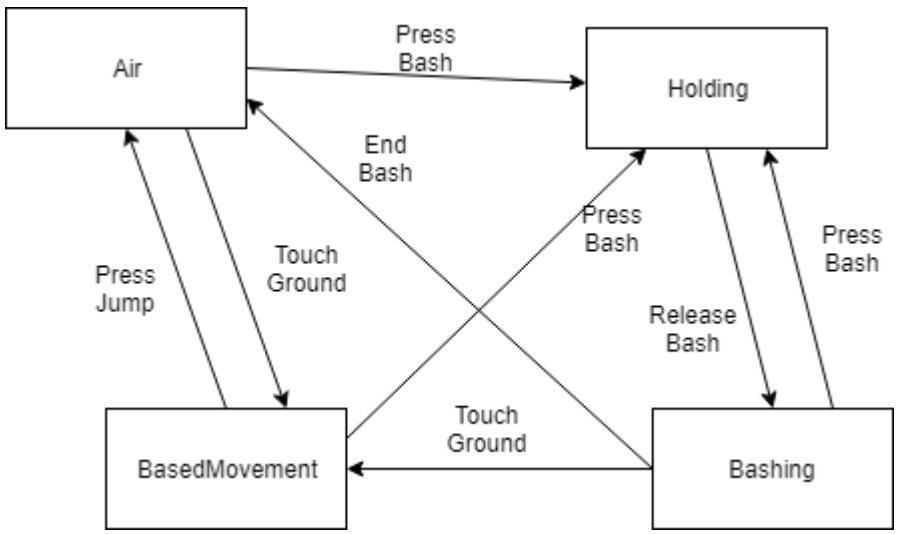
To implement an Ori like bash mechanic the essential anchor point is the player character. In the source it exists in from of a BashPlayerCharacter_BP blueprint. Besides the character, bash points, which the character can bash from, must be marked as such. This happens in form of a Bashable_BP Blueprint Interface. The Bashable_BP Blueprint interface provides methods the BashPlayerCharacter_BP Blueprint needs to interact with bash points. In the prototype there are to bash points which implement Bashable_BP, the BashBall_BP and Sphere_BP. The Goal_BP Blueprint is a lightweight goal, which just acts like a trigger teleporting the ball in the center when overlapping occurs.

BashCharacter_BP (Blueprint - Actor)

The character isn't moving under physics simulation. He is moved by code entirely. The Actor consists of a capsule as the root component, a skeletal mesh and a particle system. The blueprint implements a state machine design which is achieved by a switch construct and an enum field State. Depending on which state is active an according method is invoked. All these methods a preceded by Update, but not all methods preceded by Update are state methods. The Tick Event looks like this:



In total there are 4 states the character can be in. Air, BasedMovement, Holding, Bashing. Here is a diagram describing its functionality:



OnComponentBeginOverlap (Capsule) (Event)

When an actor begins to overlap the capsule and implements the Bashable_BP interface it's added to the BashObjects array.

OnComponentEndOverlap (Capsule) (Event)

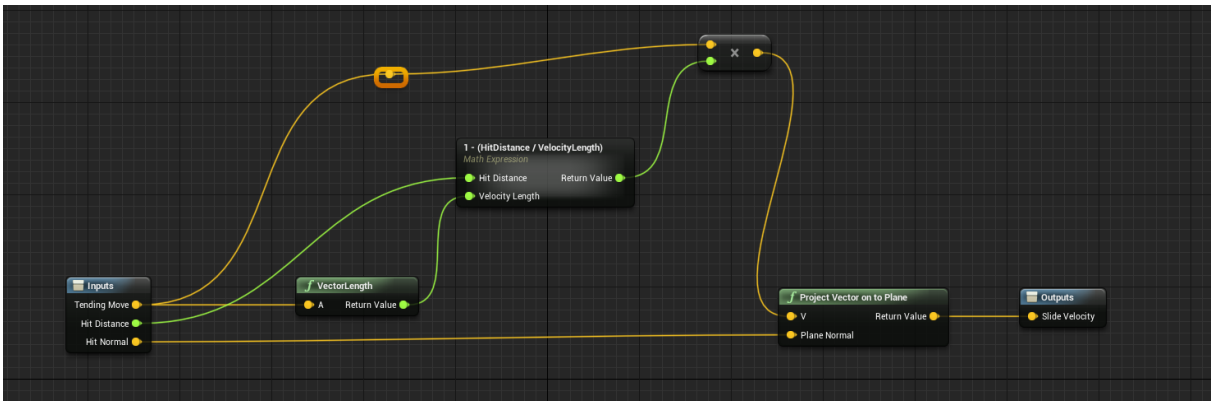
When an actor ends to overlap the capsule, it's removed from the BashObjects array.

AdjustFaceDirection (Method)

This method turns the character in the right direction depending on Velocity's X. A small dead space between -0.001 and 0.001 does nothing to prevent unwanted turning due to controller preciseness.

MoveAndSlide (Method)

This method uses a recursive approach to calculate target position giving by the characters Velocity. The character is moved by the AddActorWorldOffset method and on hit a point constrained movement is calculated. This calculation looks like that:



Where Tending Move is the move that's desired, Hit Distance is the distance being traveled until hit and hit normal the normal of the surface being hit.

This calculated movement is feed into MoveAndSlide again until no hit is detected anymore. When the player hits a surface with a specific slope angle, it's taken as a ground and the player is transferred into BasedMovement state.

UpdateClosestBashSocket (Method)

This method goes through BashObjects, which were collected by overlap events. It sets PrioritizedBashable to null. For every object it tests whether it has a higher priority. First it takes the first bashable object. It uses the IsBashable method of the Bashable_BP interface to get that information. Afterwards every bashable object with higher priority, which is given by the method GetPriority of the Bashable_BP interface, or same priority and closer location is set as PrioritizedBashable.

UpdateBashIndicator (Method)

This method will set the target parameter of the particle system to the PrioritizedBashable's Location. When no PrioritizedBashable is found, it'll set the particle system to invisible.

BufferBashDirection (Method)

This method buffers the normalized input of the left analog stick in Direction for how long BashDirectionBufferTime says.

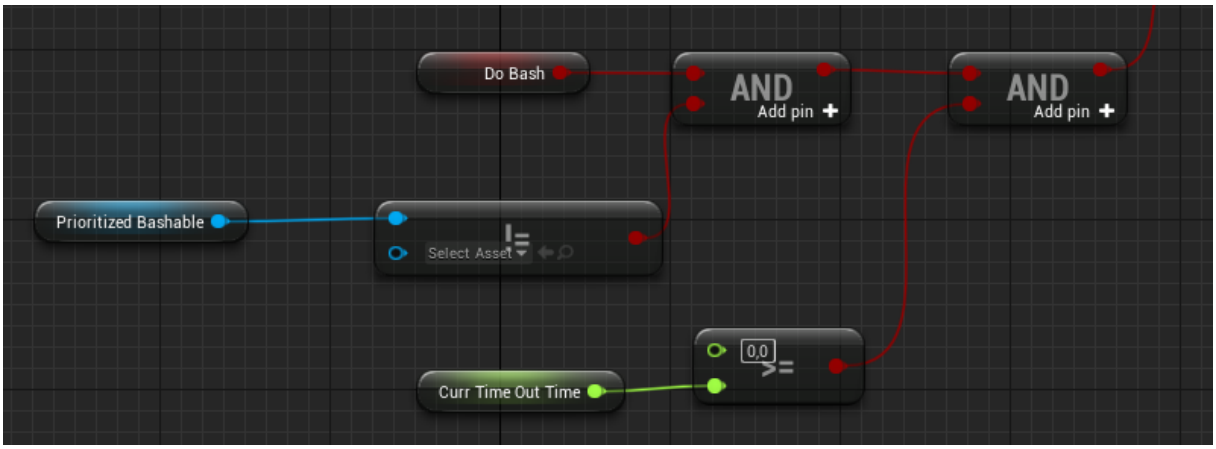
UpdateTimeOutTime (Method)

This method counts the CurrTimeOutTime variable down. This way it can be set to give the character a time out for bashing.

State machine

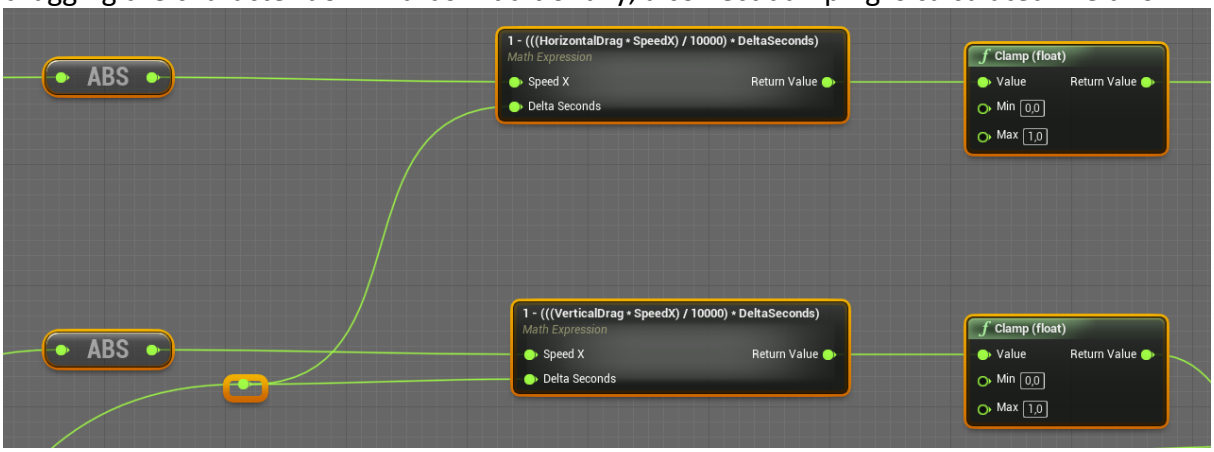
BasedMovement state

In the BasedMovement state the character is instantly moving by input of the left analog stick. The character is moving linearly to the input giving until MaxWalkSpeed is reached. A walk animation through the animation blueprint is played. When the character starts bashing RegisterBasher is invoked on the prioritized bashable which returns a BashSocket blueprint for communication. At the beginning the TimeOutTime is updated, which makes sure that after a player falls of a bash point he cannot bash directly again. Condition for bashing are: a PrioritizedBashable exists, the player wants to bash and the TimeOutTime is below zero.



Air state

In the Air state the character accelerates with the input of the left analog stick. Gravity is dragging the character downwards. Additionally, a correct damping is calculated like this:



The character can bash in Air state as well. MoveAndSlide is used for movement as well.

Holding state

In the Holding state the input of the left analog stick is buffered by the method BufferBashDirection. When the input is a zero vector the previous input is buffered for BashDirectionBufferTime. When the buffered players input is a zero vector the character is transferred into Air state and just falls of again. When the buffered players input is not zero the character is transferred into Bashing state. Additionally, the character falls of the bash points and is set into Air state after BashTime has run out. The character will get a time out for bashing as well so that the character can't bash again immediately.

Bashing state

The Bashing state linearly moves the character along the BashDirection until the BashPlayTime runs out. After that, it applies a velocity in the BashDirection and transferring it back in Air state. While Bashing the character can immediately initial a bash again.

Bashable_BP (Blueprint interface)

This is a blueprint interface used to mark blueprints as being bashable. Most of the functions are used by the BashCharacter_BP to gather information about the bashable. However, RegisterBasher starts an interaction. It returns a BashSocket which is used for communication between bashable and character.

BashSocket (Blueprint - Actor Component)

The BashSocket is used for communication between bashable and character. The variable ArrowComponent and Basher must be set at initialization. BashStarted can be bind by the actor to react to bashing. The SetBashDIrection function will rotate the Arrow in the according direction.

Sphere_BP (Blueprint – Actor Component)

RegisterBasher

When RegisterBasher is called, a BashSocket is initialized with a new ArrowComponent. It'll bind the BashStarted event. When the BashStarted event is called the BashSocket and ArrowComponent get deconstructed.

BashBall_BP (Blueprint – Actor Component)

The BashBall is Physic simulated. The Physics Material is BouncyObject, which is just a 100 % bouncing material.

RegisterBasher

When RegisterBasher is called, a BashSocket is initialized with a new ArrowComponent. It'll bind the BashStarted event. Additionally, Physics simulation will be stopped. When the BashStarted event is called the BashSocket and ArrowComponent get deconstructed, the BashBall gets an impulse in the direction of BashDirection and physic simulation will be back on.