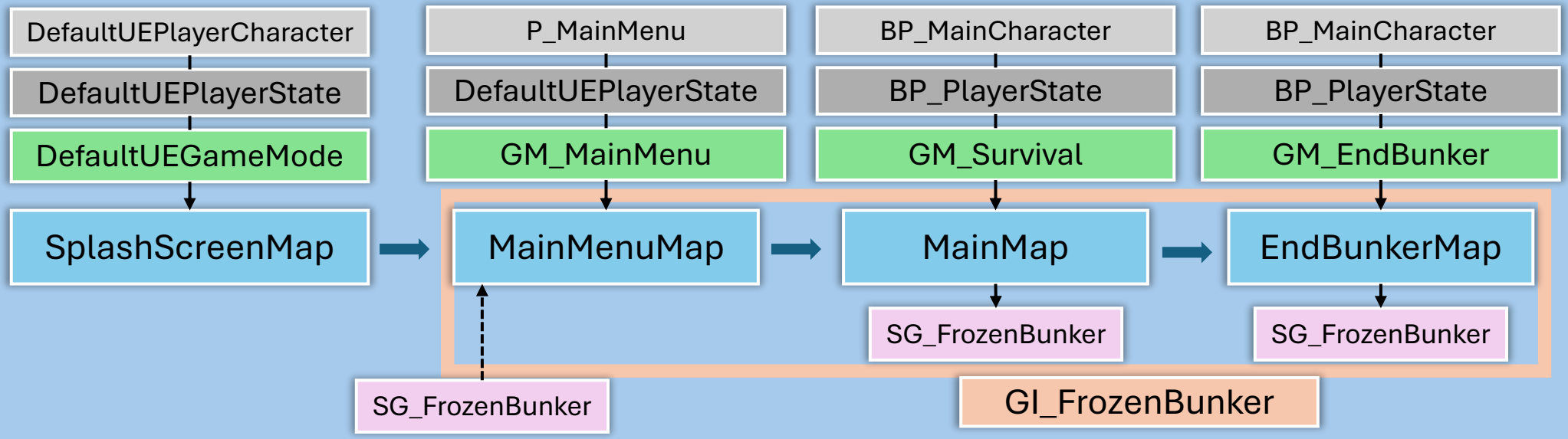
FROZEN BUNKER

TECHNICAL DESIGN DOCUMENT

BONNET Tristan

FROZEN BUNKER

GAME ARCHITECTURE



GAME MODE

Each **scene** is linked with a **specific GameMode**, that **manages what will happen** in the scene itself.

GAME INSTANCE

To **save variables among each scene** I use the GameInstance.

The only **variable** a need to transfer among level is the **playerName**.

PLAYER STATE

To **manage variables** linked to my **player** I use the **PlayerState**.

PLAYER CHARACTER/PAWN

I use the PlayerCharacter / Pawn just to **receive input** and other **basic stuff**.

SAVE GAME

The save game to **stack the variable locally** on the **computer**.

In my case, I **stack a specific list of structures** that appear in the **MainMenuMap**.

SETUP SYSTEM

GM_Survival

- The setup is made by the main GameMode : **GM_Survival, on the MainMap**.

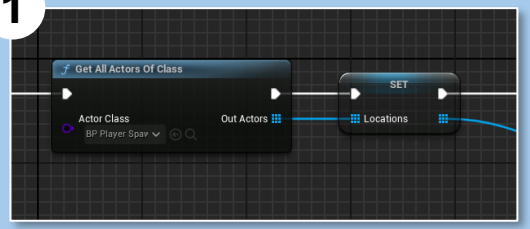
PLAYER RANDOMIZATION STARTING LOCATION

1. The GameMode gets **all the possible locations** where the player can start, those locations where **symbolized by a special blueprint** class called : **BP_PlayerSpawnLocation**.

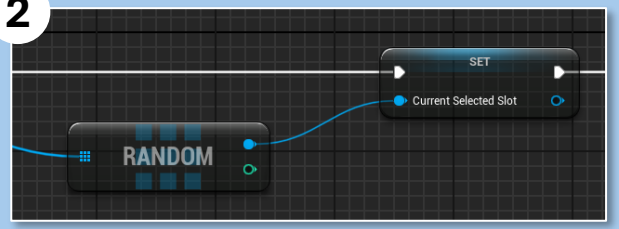
2. Once the GameMode has the list, the system **gets a random item in this list**, a **line trace** is made **from the item location to the ground**. Player is **put at the impact** location, for the **rotation** he takes the **one from the item**.

3. Next, to put the **tent in front of the player** : a **trace** is made from a **point in front of him** to the **bottom to hit the ground**. The tent location is at the impact point and the **rotation** is made by **looking at the player**.

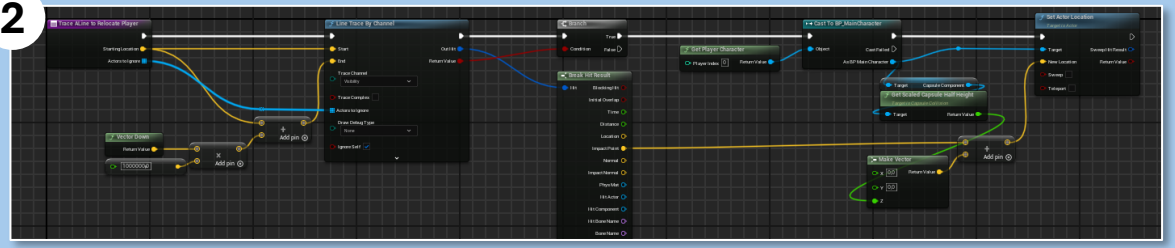
1



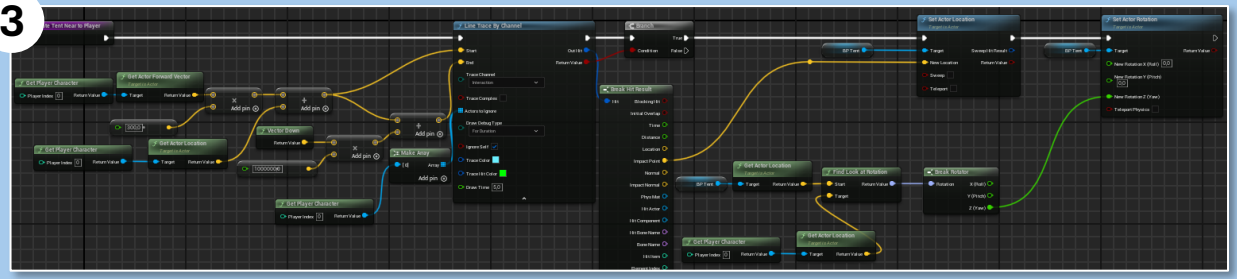
2



2



3



SETUP SYSTEM

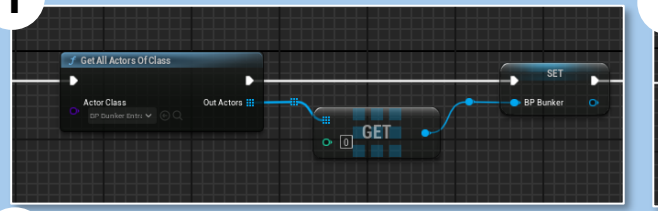
BUNKER ENTRANCE RANDOMIZATION LOCATION

1. The GameMode takes the **reference** of the bunker entrance **: BP_BunkerEntrance**.

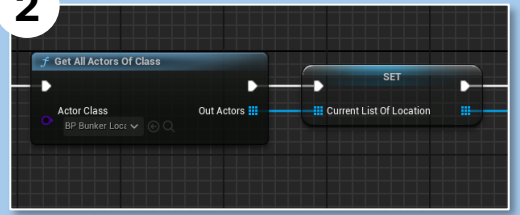
2. The GameMode gets **all the possible locations** where the entrance can be, those locations where **symbolized by a special blueprint** class called : **BP_BunkerLocation**.

3. Once the GameMode has the list, the system gets the **distance between playerCharacter and each bunker location**. Next the system **selects randomly** one of the three **farest bunker locations** gets the **location and rotation** of the selected BP_BunkerLocation and **set them to the BP_BunkerEntrance**.

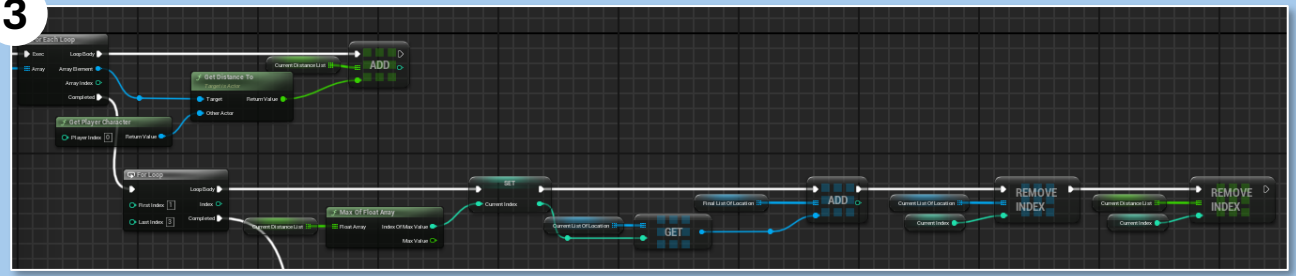
1



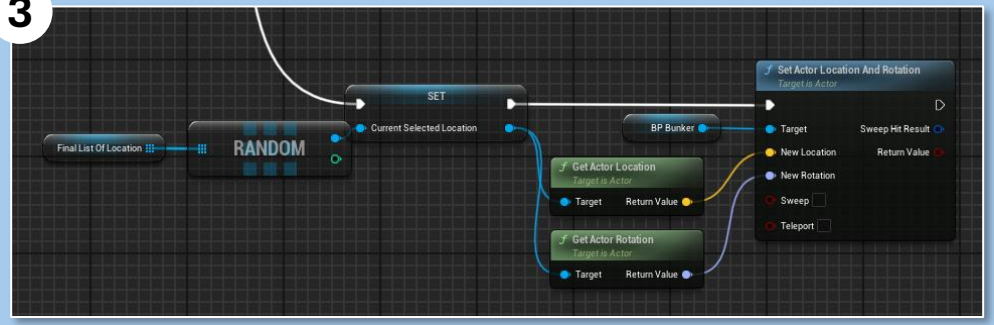
2



3



3

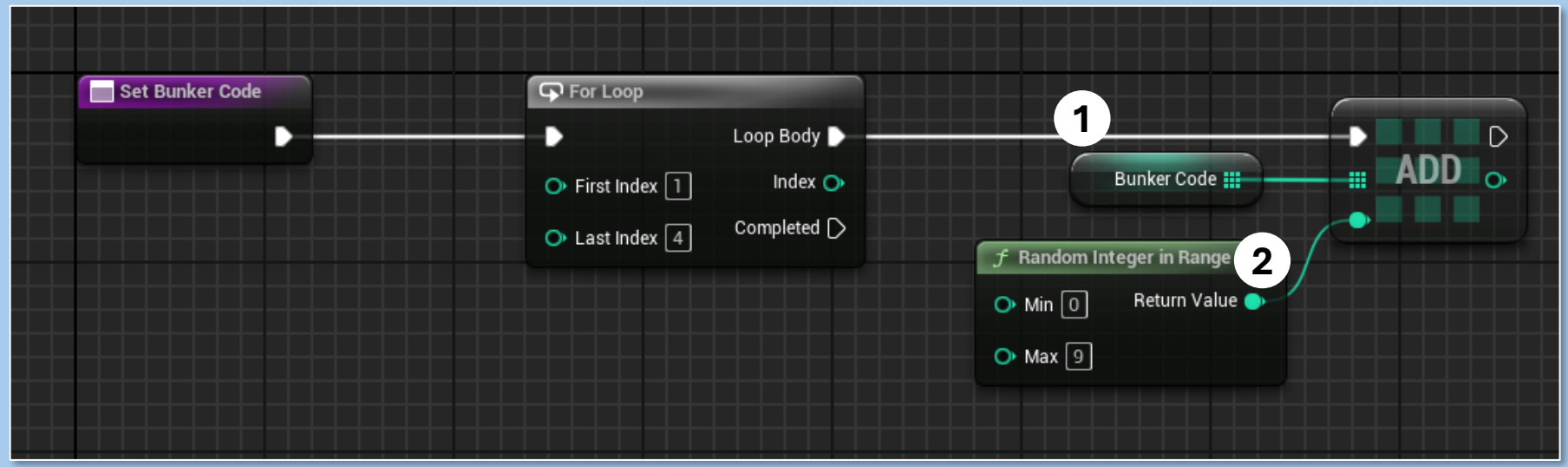


SETUP SYSTEM

BUNKER CODE RANDOMIZATION

1. The bunker code is saved in a **4 items list of integer**.

2. Each item is set by **taking randomly** an integer between **0 and 9**.



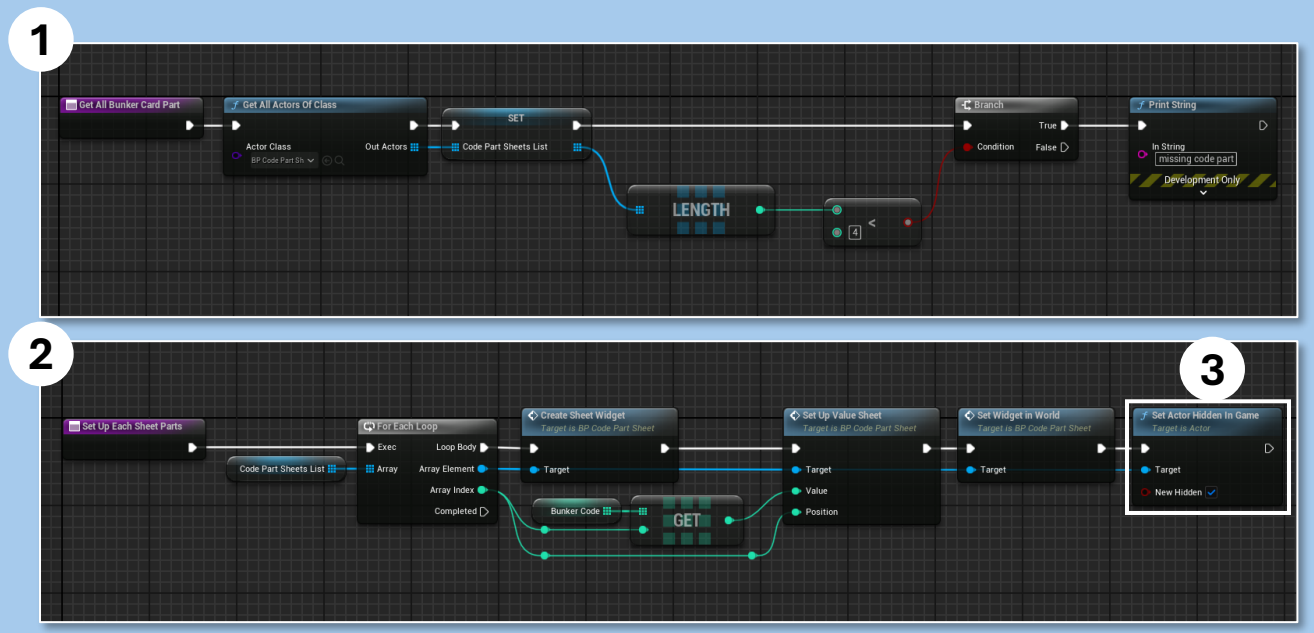
SETUP SYSTEM

BUNKER CODE PARTS - INITIALIZATION

1. Code parts are **actors that show the code to the player**, they are a class of : **BP_CodePartSheet**. There are **4 code parts on the map** (one for each code number). At the beginning of the game all the code parts are **got by the GameMode**.

2. The first step is the initialization : **each BP_CodePartSheet will receive a number** of the bunker code, the number will be used to **update a widget** present in the blueprint.

3. Each code part is **not visible** in the world at the **beginning of the game**, it will be visible later.



SETUP SYSTEM

BUNKER CODE PARTS – LOCATION RANDOMIZATION

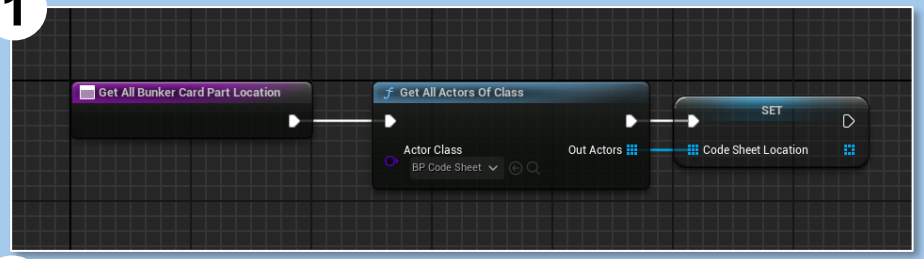
1. Bunker code part locations are obtained from a blueprint : **BP_CodeSheetLocation**. This blueprint contains one **integer variable that symbolizes the area** where this blueprint is. All those blueprints are **got at the beginning of the game** and put in a list.

2. When the **player interacts** with a special actor : **BP_Message2** (this interaction means the player found the bunker entrance) **each code part can be relocated**.

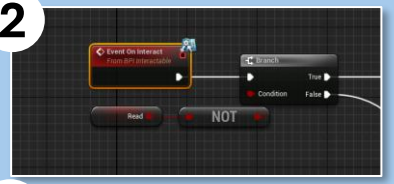
3. For the relocation, the system will **select randomly a BP_CodeSheetLocation** in the list, **put a code sheet at this location**, with this **rotation** and **set it visible**.

4. Then all **the BP_CodeSheetLocation with the same area index will be removed** and the operation is repeated for each code part.

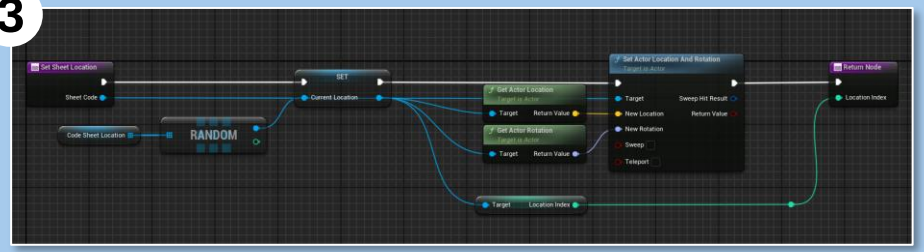
1



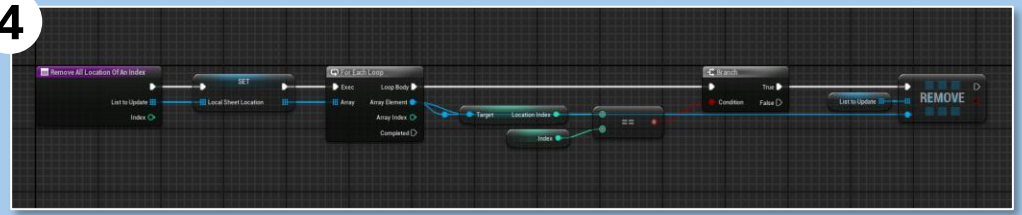
2



3



4

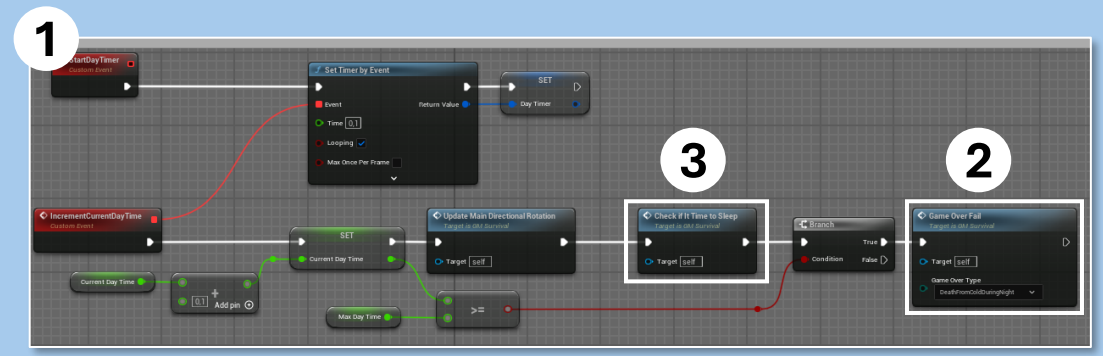


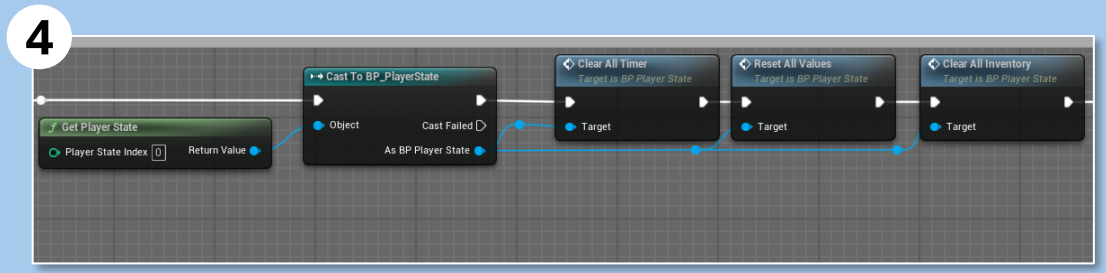
FROZEN BUNKER

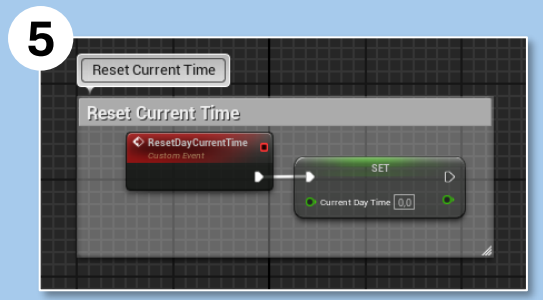
DAY SYSTEM

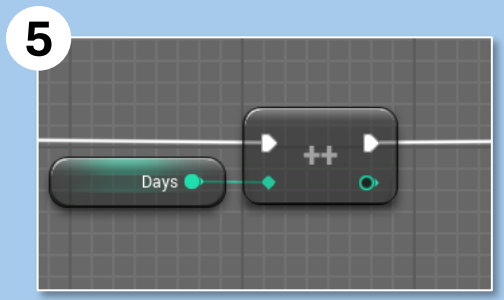
Day system is managed in the **GM_Survival**.

1. A **timer is set** when the gameMode is started and the variable : **currentDayTime is incremented**.
2. When this **value is equal to maxDayTime**, the **game is over**.
3. But the player **can interact with the tent** when this **value is equal to or higher than minTimeToValidDay**.
4. When the **player interacts with the tent**, all player **statistics are reset** and all the **inventory is cleared**.
5. Then the **currentDayTime** value is **set to 0** and the **days value** (symbolized by an int) is **incremented**.









FROZEN BUNKER

PLAYER STATISTICS

Player statistics are in the BP_PlayerState.

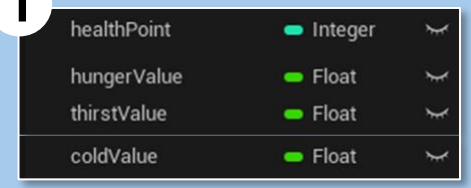
1. There are the **health**, the *hunger*, the **thirst** and the **cold**.

2. **Health** is symbolized by an **int value**, when it reaches **0**, the **game is over**.

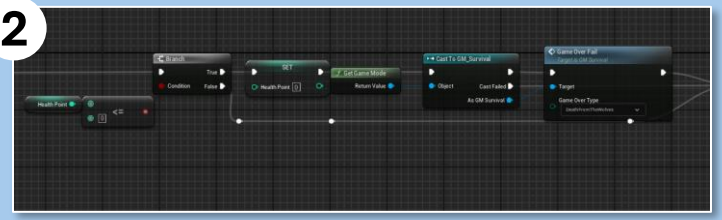
3. **Hunger** and **thirst** are symbolized by **float values** that **decrease with time**, when one of those values **reaches 0** the **game is over**.

4. The **cold** is symbolized by a **float value** that **increases with the time**, when this **value reaches timeToDieFromCold** (set to 180 sec) the **game is over**.

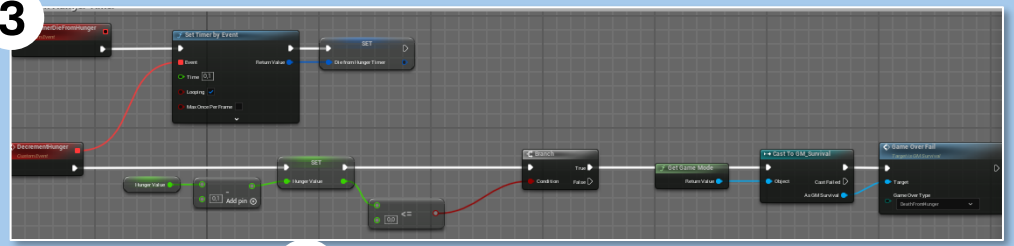
1



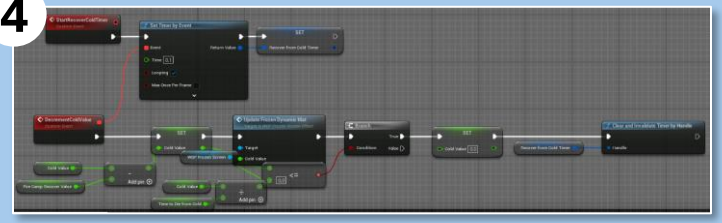
2



3



4



FROZEN BUNKER

INVENTORY SYSTEM

INVENTORY CLASS

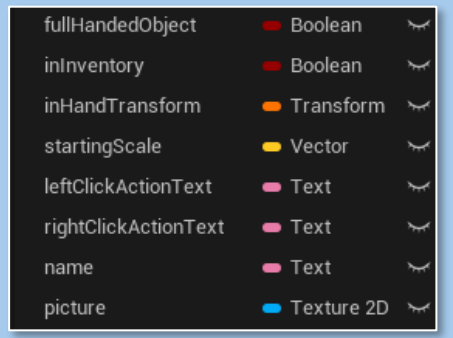
- All inventory items **derive from** a parent class : **BaseClassInventoryObject**, this class has some **variables**.

fullHandedObject: specific **object that takes all the place** in hand of player (in my case fullHandedObject is the **fire camp**), when this **object is in hand the other object in hand is automatically disabled**.

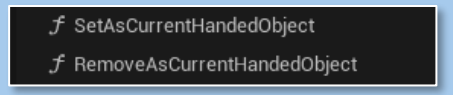
inHandTransform : the **transform** item will have when it is **snapped to the player.**

startingScale : the **scale** object has at the **beginning of the game**, when the **item is dropped** by the player it will **recover this scale**.

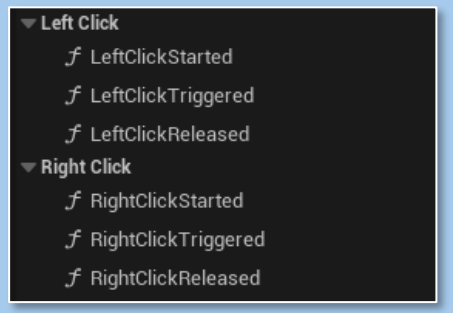
Other **variables** are here for the **user interface**.



Those **functions** are used when the item is **set as the current object** and the **other when is removed**.



Those functions are used to **use an item when** it is set as **currentHandedObject**.



These **functions** are called in the **playerCharacter.**

FROZEN BUNKER

INVENTORY SYSTEM

INVENTORY SETUP

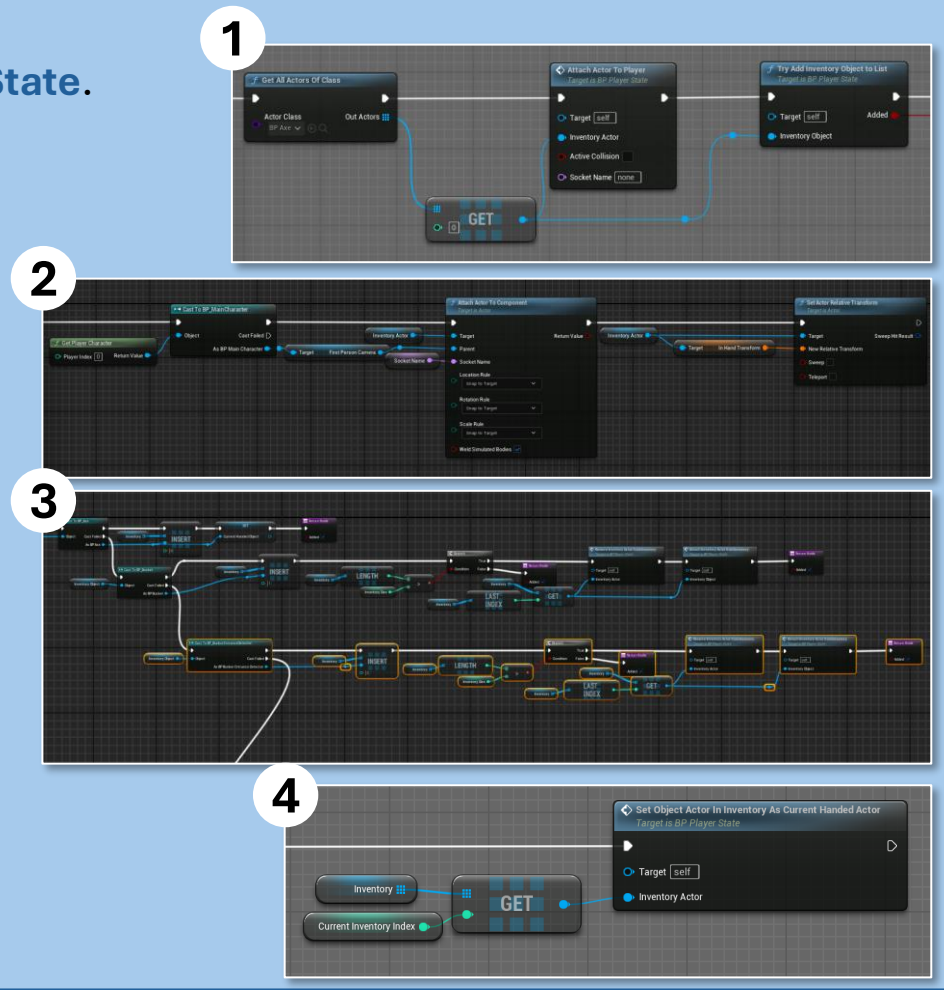
Inventory is an **array of BaseClassInventoryObject** stack in the **BP_PlayerState**.

1. At the **beginning of the game**, the **mandatory objects** (axe, bucket and detector) are **automatically added to this array**.

2. Then they are **attached to the player** and **use the inHandTransform** to set their **transform when they are attached**.

3. **Inventory items** are added in **this order** :
    1. Axe index : 0
    2. Bucked index : 1
    3. Detector index : 2

4. At the **beginning of the game** : the **axe** is set as the **currentHandedObject**.

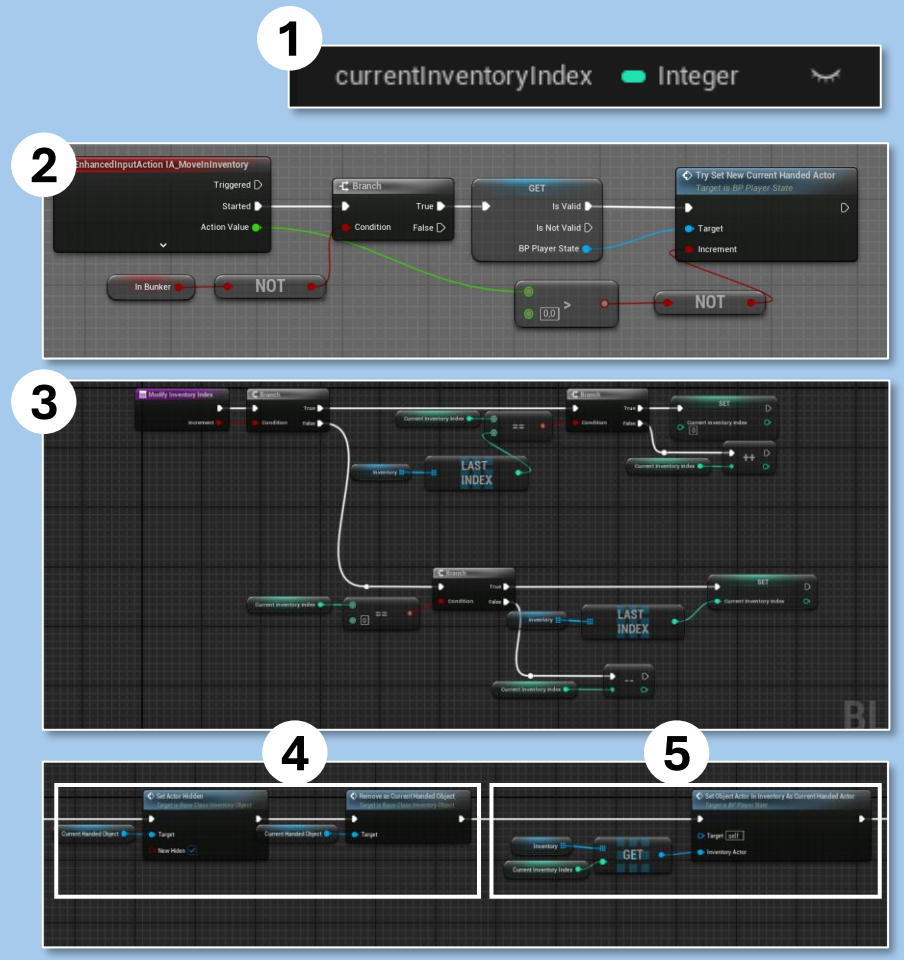


FROZEN BUNKER

INVENTORY SYSTEM

MOVE IN INVENTORY

1. To **move** in inventory, a variable **inventoryItemIndex (int)is used**.
2. The function to **modify the index** is called in the **BP_MainCharacter**, by **using wheel up/down**.
3. In function of the **input received,** the **index is modified**.
4. Then the **last handed object is disabled** (visibility is set to false).
5. In the **inventory array**, the **item at the new index value is taken** and set as the **currentHandedObject**.



FROZEN BUNKER

INVENTORY SYSTEM

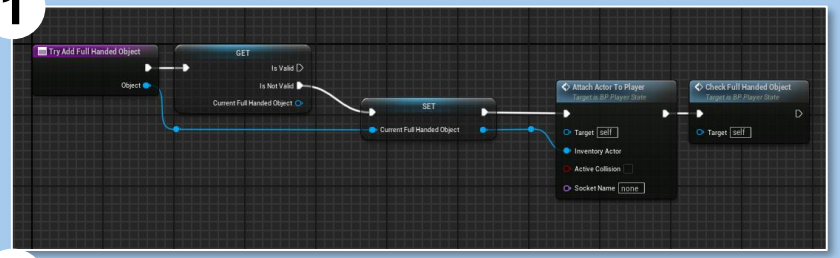
FULL HANDED OBJECT - WOOD CAMP

1. When **a currentFullHandedObject is trying to be set** the system checks if **there is no currentFullHandedObject**, if not the **variable is set with the object created**.

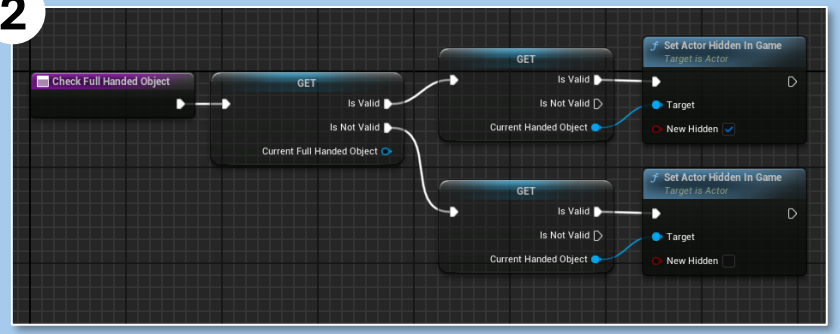
2. The **currentHandedObject is hidden**.

3. When the **currentFullHandedObject is finished to use** (in the case of the wood camp just putted on the ground), the **fullCurrentHandedObject variable is set to null** and the **currentHandedObject is set visible**.

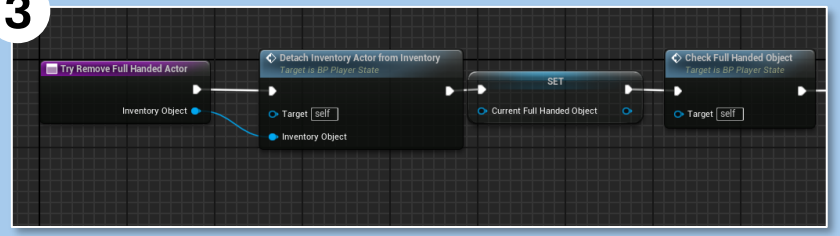
1



2



3



WOLVES

SPAWNING SYSTEM

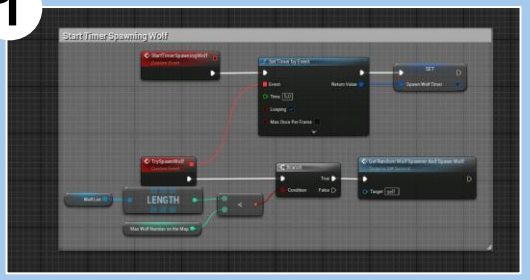
The **game mode** manages **wolves spawn**.

1. A **timer is set**, and **each time**, the event to **spawn one wolf is called**.

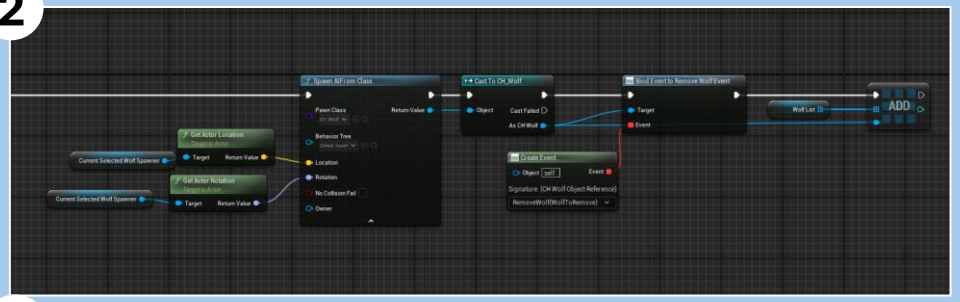
2. During this event, the **system will select randomly** an actor in the list of **BP_WolfSpawnerLocation**, and **spawn the wolf** a this location, the **wolf is added to a list**.

3. Then the **actor is removed from the list** and when **all actors are removed** the array is **refilled with all BP_WolfSpawner location actors**.

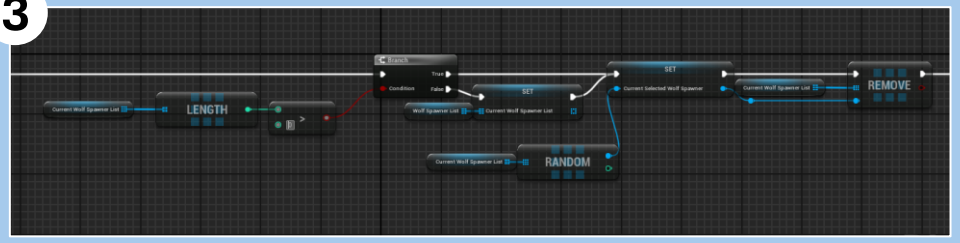
1



2



3



WOLVES

WOLF BEHAVIOR

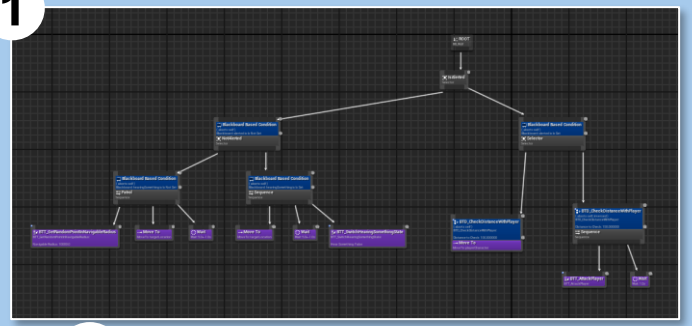
Wolf behavior is managed by a **behavior tree**

1. A **wolf** has **two main states** : **patrol** and **alerted**.

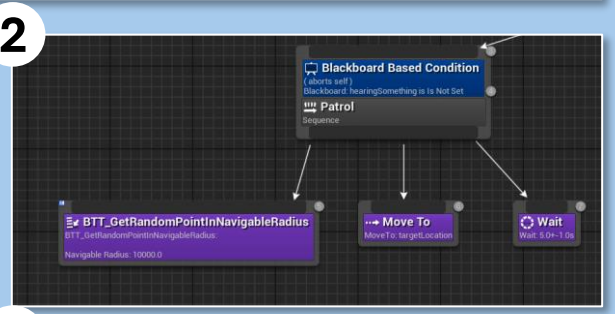
2. When **the wolf is in patrol** state it will take a **random point** in a radius around him**, move to this location**, **wait a time** and redo this action.

3. When the **player runs**, he **makes noise**, the **location of the noise** is the **new target location** in their patrol.

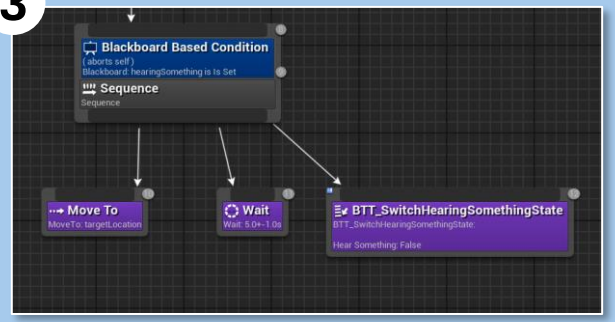
1



2



3



WOLVES

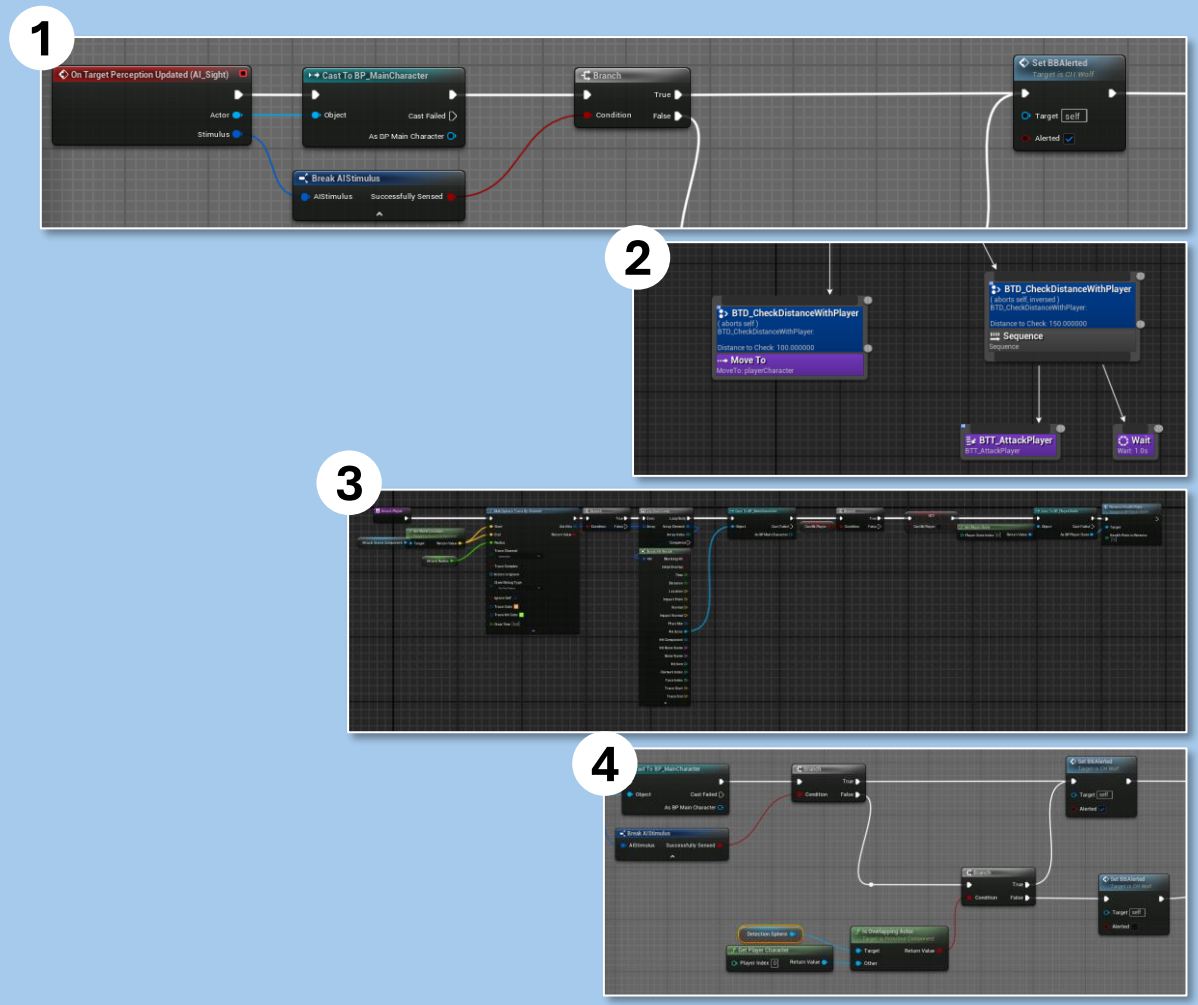
WOLF BEHAVIOR - ALERTED

1. When the **wolf sees the player it switches to alerted state**.

2. In this **state if the wolf has player in sight** it will **move to him**.

3. When it is in **front a player**, an animMontage is played and the **function Attack Player is called**.

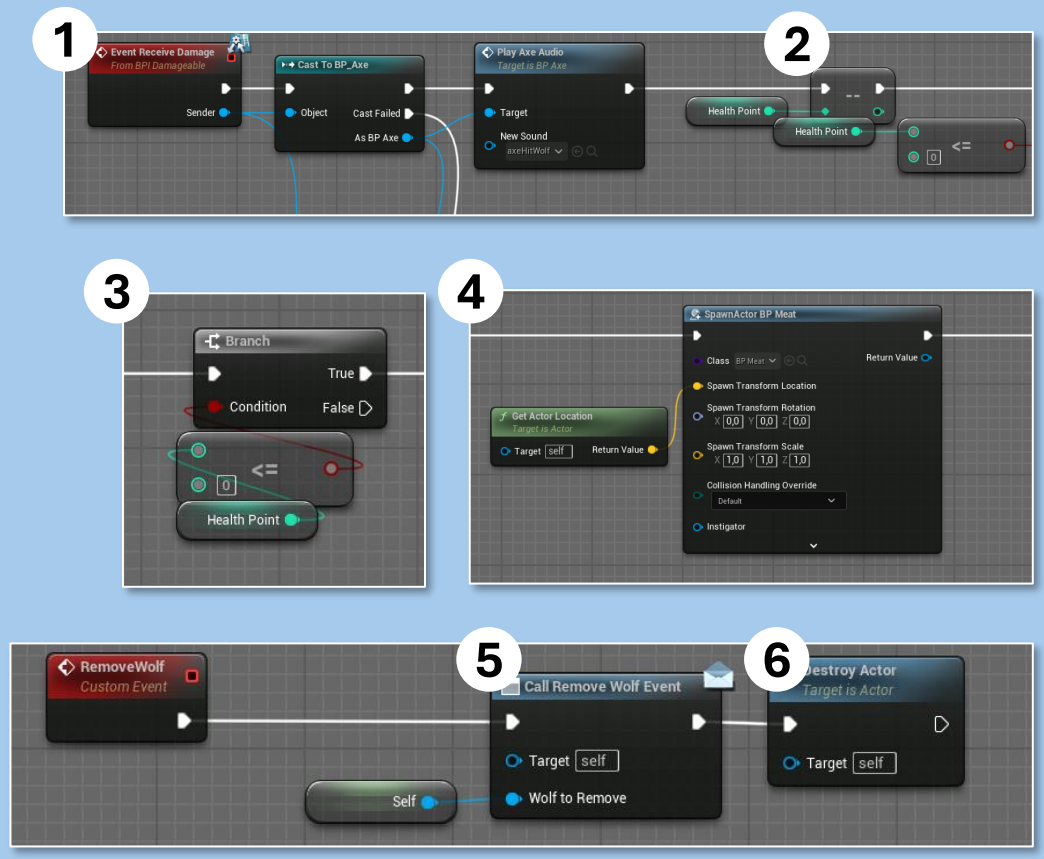
4. If the player **leaves the sight of the wolf**, the system **checks if the player is in the wolf area** (symbolised by a sphere component) if **yes the wolf is still in the alerted** state. If **player is not in the area** the wolf will **return in patrol state**.



WOLVES

WOLF DEATH

1. A **wolf takes damage** the **axe overlaps him** during his used.
2. An axe shot **removes 1 healthPoint** to the wolf.
3. When wolf has **0 health point** the **wolf is dead**.
4. A piece of **meat spawns**.
5. An event is **called to the game mode** to **remove the wolf from the list**.
6. The wolf **actor is destroyed**.



FIRE CAMP

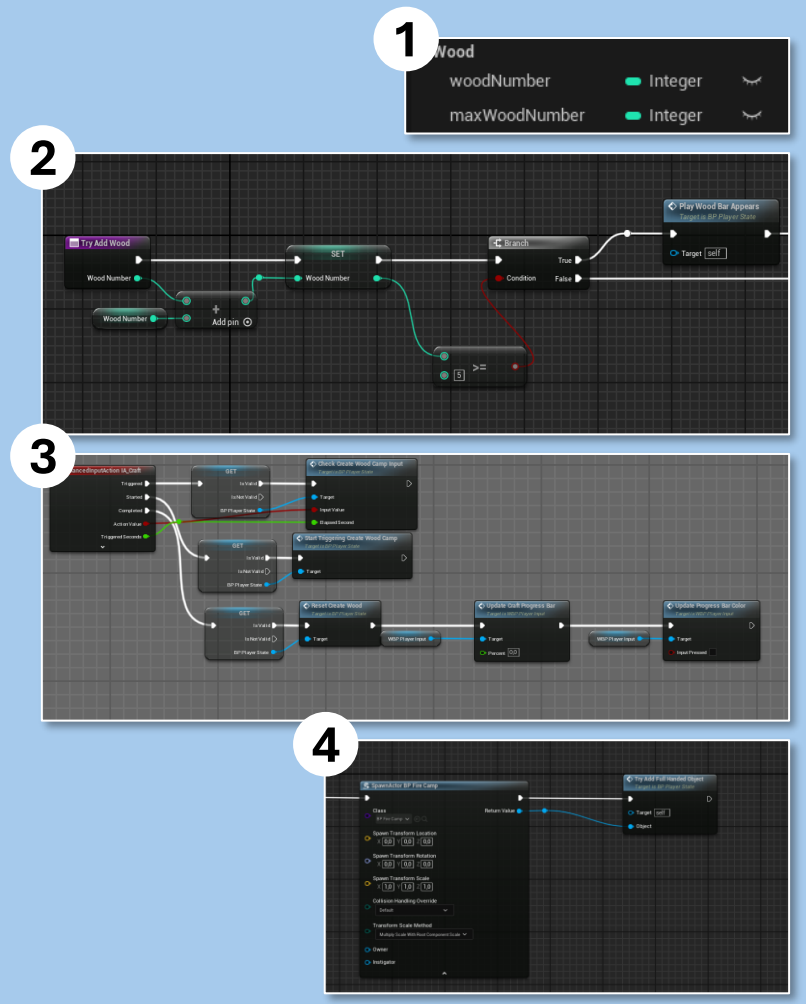
CREATION

1. A **variable wood** is stacked in the **BP_PlayerState**, this variable **cannot be higher than 20**.

2. When the variable is **equal to or higher than 5**, the player can **create a fire camp**.

3. He needs to **maintain the « A » key** for a certain time.

4. Then the fire camp actor spawns and **his automatically set has the currentFullHandedObject** in the BP_PlayerState.



FIRE CAMP

USE

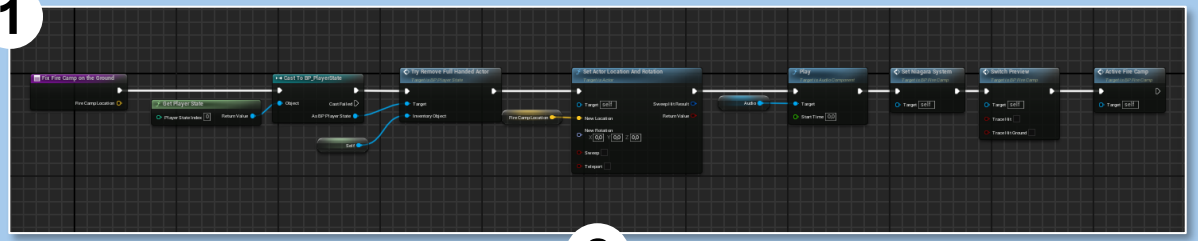
1. When fire camp is **in player hand**, it can be **put on the ground** with left click.

2. When the fire camp is on the ground a **timer is set to decrement the lifetime value**.

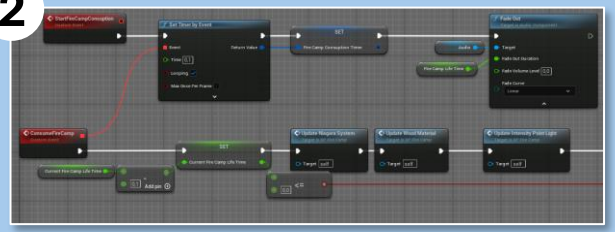
3. When the value is **equal to 0**, the **fire camp is disabled**.

4. During the **fire camp lifetime**, if the **player stays in the area** (symbolized by a sphere component) his **cold value decreases**.

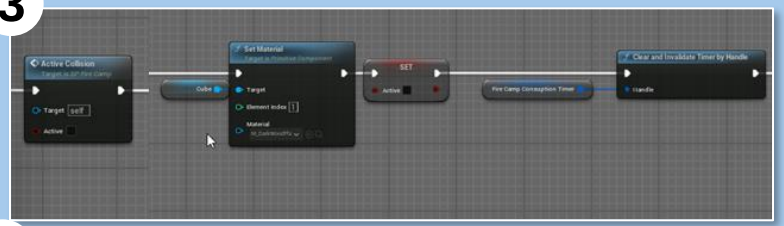
1



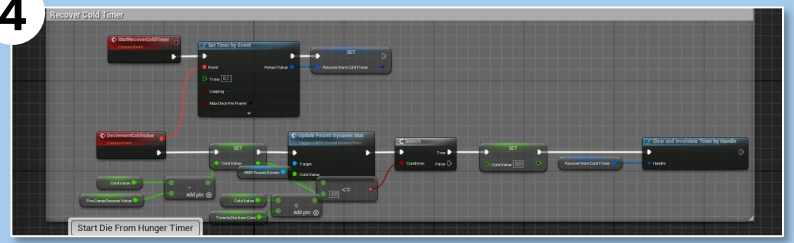
2



3



4



FIRE CAMP

COOK

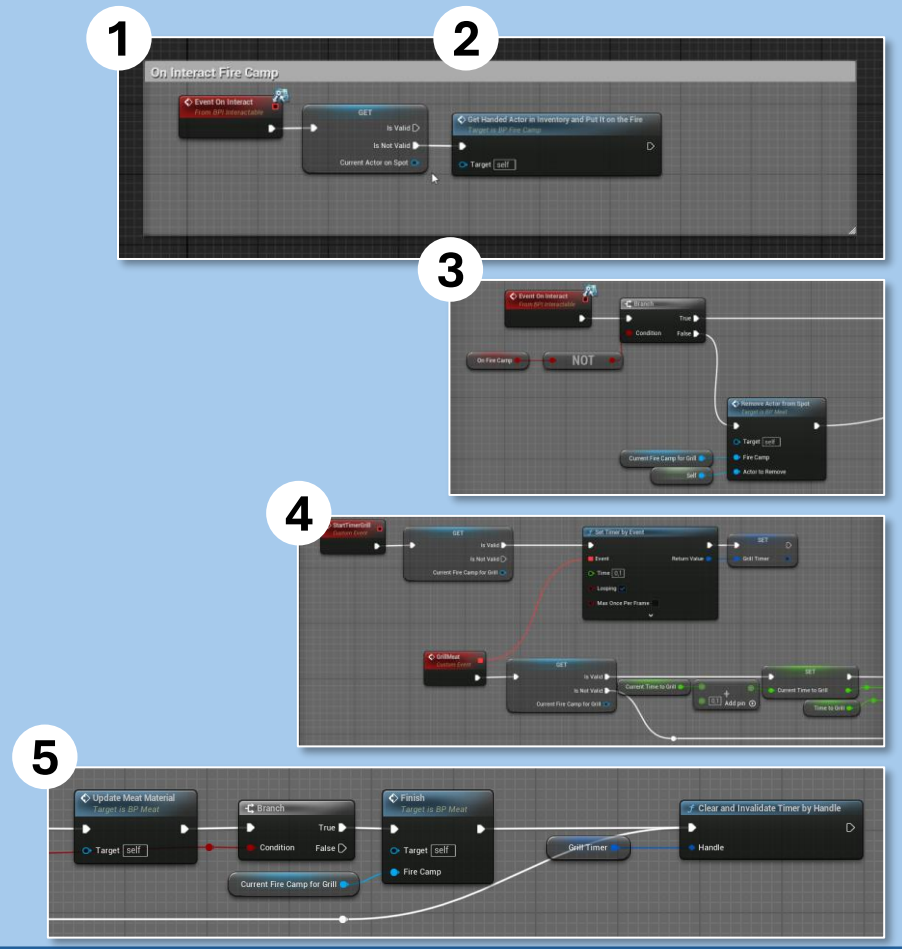
1. When fire camp is **put on the ground**, the player can put a **piece of meat or the bucket on the camp**.

2. When one of those objects is **put on the fire camp** the **slot is not available**.

3. The **object can be removed** from the fire camp **at any time**, then the **slot is available again**.

4. When the object is **on the fire camp**, an **internal timer starts to cook the object**.

5. When the **object is cooked**, it **will stay on the fire camp** and the **player needs to interact** with to **remove** it from the fire camp.



END GAME STATISTICS

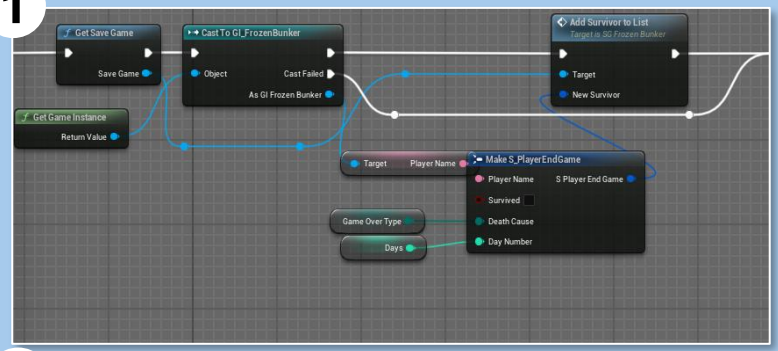
BEHAVIOR

1. At the **end of the game** (fail or success), **informations** of it are stack **in a structure** : **S_PlayerEndGame.**

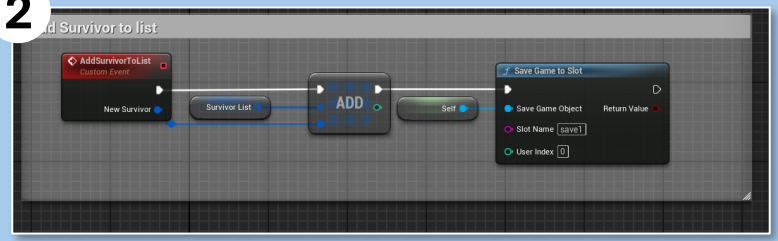
2. This structure is **saved in an array contained in the SaveGame** : SG_FrozenBunker.

3. When **the MainMenuMap is open**, the system **gets the array and creates one widget for each structure** to **show the history** of the games.

1



2



3