Background

<<ORIGIN>>

<<Insert tragic backstory here>>

Profile

Name: Osakabe

Species: Humanoid(-ish?)/Yokaï

Occupation: She's not seen much, said to only leave her chambers once a year. No one quite knows what goes on behind the doors the rest of the year, she isn't actually the social type..

Gameplay

DON'T LOOK!

She may be sly and dangerous aura seem like something you want to keep an eye on. Resist that urge, it'd be the last thing you'll ever see!

Her Lovely Pets

Who doesn't like a friendly fox?

Physical description

Slim with Hunchback Posture

- **Face:** She wears a mask, not many know what's underneath...
- **Height:** 5'9", but her 12 layered Kimono makes her seem a lot bigger!

Oh, She Has 9 Arms

- How does she use them?

Use This Document

Click **middlemouse** to drag page around!

Press **Z** to zoom out!

BEHAVIOUR

In General

The AI is executed through behaviour tree but in concept it is closely related to finite state machines. Each state has its own set of behaviour and works closely together with the detection system in regards to transitioning in between them.

I'll describe each state on a high level first and then further down you'll find visualization for each of the individual tasks.

	SIGHT	VISIBLE HEARING	INVISIBLE HEARING	TOUCH	Completed TASK
DEFAULT	↓	↓	↓↓	↓↓↓	
SUSPICIOUS	↓		↓	↓↓	↑
INVESTIGATE	↓	↑		↓	↑↑
ALERTED					↑↑↑

RESEARCH

Behaviour Systems

Default

During this state the Osakabe will roam the halls of the castle in search of the player. It will move to and around the objective area and it's surrounding points of interest.

During this state the AI isn't at all aware of the player.

Suspicious

When the AI has heard something and visibility to it's origin isn't blocked the AI will go into the suspicious state. The AI will turn and look towards the perceived origin of the stimuli. After looking it will either detect the player or go back into the default state if nothing is seen.

The AI does at this point have a 'soft' reference to the player. This means that if the player is seen and walks away the AI's gaze will follow the player and not just stare with an empty gaze to the stimuli origin.

Investigate

The Investigate state is triggered by the AI catching enough of a glance of the player or by a noise event that's blocked in visibility.

In the investigate state the AI will move towards the stimuli origin and then move to one more location where the player could be hidden, around a corner for example. This is completely based on it's own senses and tests, the AI doesn't know where the player is beforehand.

If the AI doesn't find the player it will proceed in the default state.

Alerted

When alerted the AI will always be fully aware of the player's location and 'dash' in front of them. When looked at by the player they deal damage.

This state has a limit in order to control the pace. After a while it will back off into the background area in order to give to player some space to breath again.

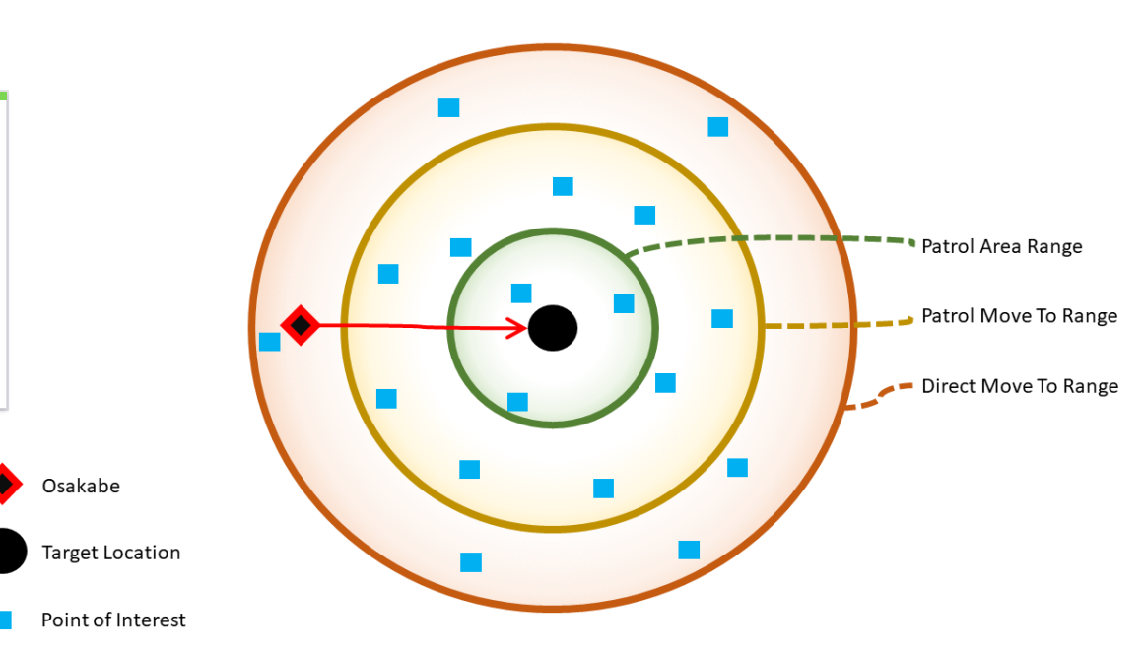
ALL SPECIFIC TASKS VISUALIZED

Default

The default state is the initial and primary state for the AI. It's behaviour is to roam around and 'search' for the player. This is broken down in several sub-branches.

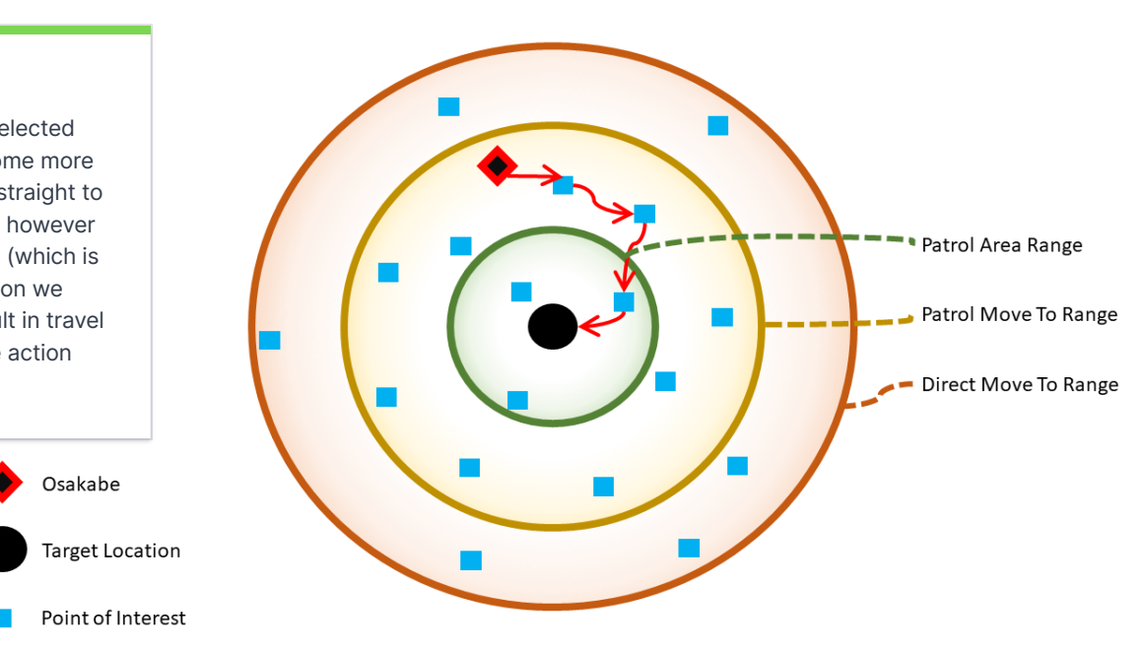
Direct Move To

The basic most movement action where the AI directly run towards its target location. The action is perserved for only long distances. As it's not the most interesting action we only use it when we need to get AI from A to B in a reasonable time-frame.



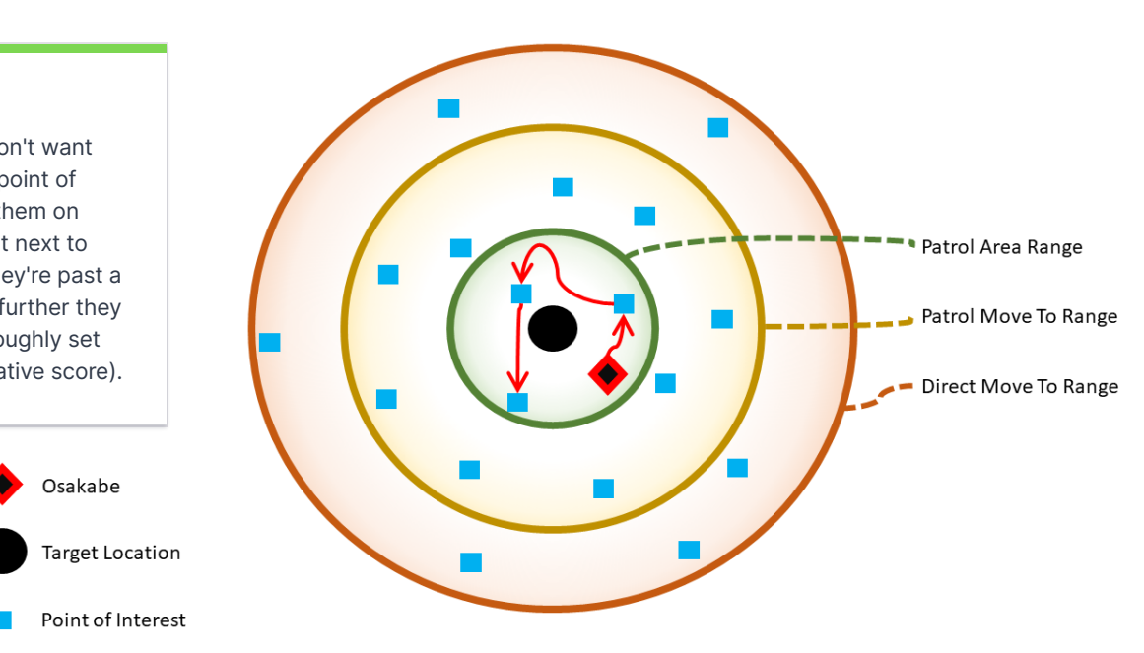
Patrol Move To

In this task the AI walks from A to B through a selected number of 'points of interest' which results in some more interesting behavior as the AI doesn't just walk straight to it's targets. This is our preferred way of moving, however it is quite a bit slower due to the inefficient path (which is on purpose to make it interesting). For that reason we don't use it for longer distances as it would result in travel times being to long and the information that the action was picked on isn't relevant anymore.



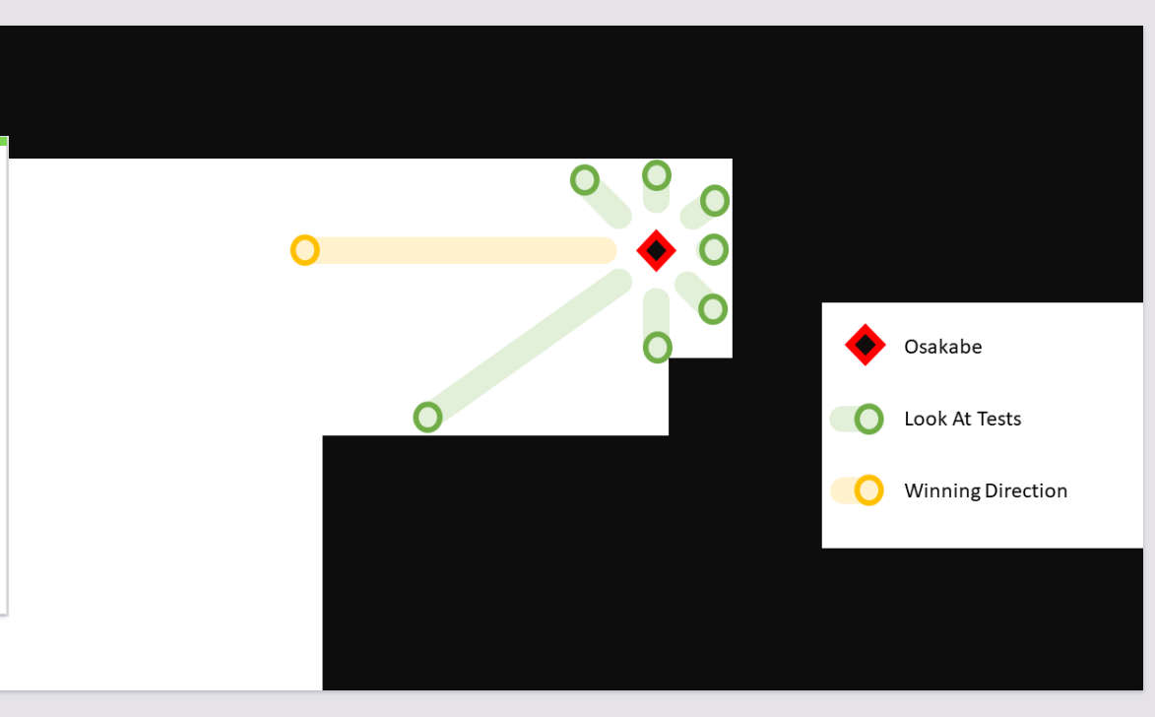
Patrol Area

When the AI is already at their destination we don't want them to just stand there. In this case we score 'point of interest' in the area to patrol through. It scores them on distance from each other (move to 3 points right next to each other wouldn't be too interesting) and if they're past a certain threshold they will score negatively the further they are removed from the target (this threshold is roughly set so points in the same room won't receive a negative score).



Look At

If a destination is reached we don't want the AI to immediately move on, we want to emphasize that moment a little bit (both as feedback but also as a window of time for the player to act while the AI doesn't move). We do this simply by make the AI scan the room for a bit. We run some tests based on where it can see the most (so it doesn't stare at a wall) and then rotate the AI to look in that direction for a while until we proceed with its following action.

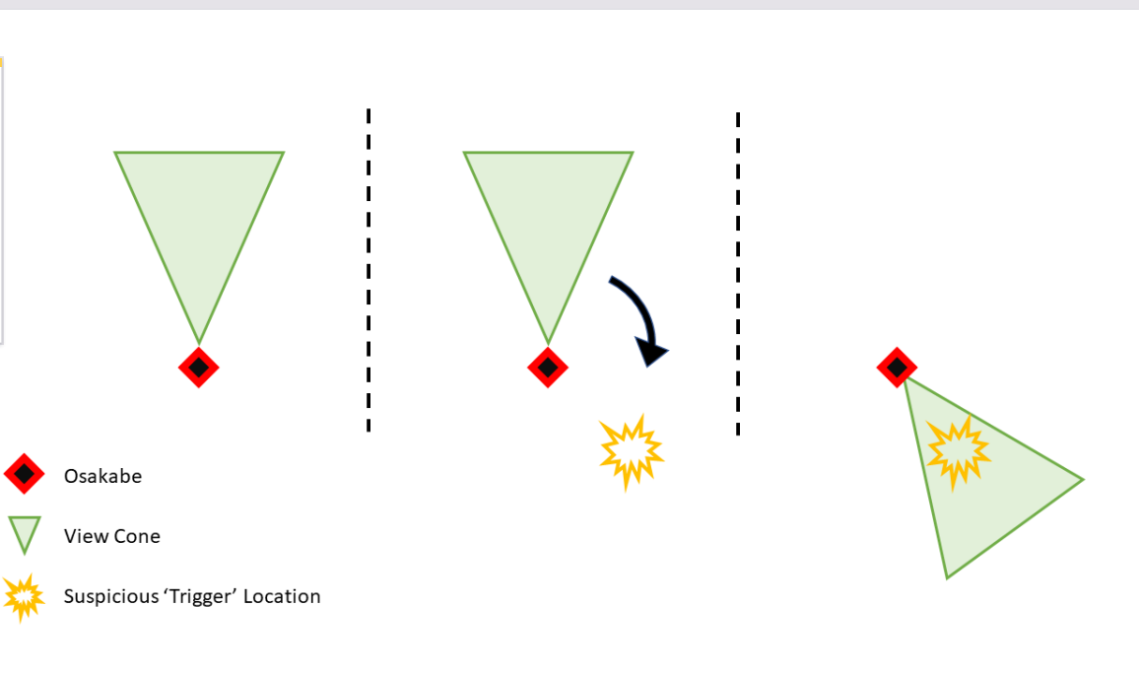


Suspicious

This state is the AI's response to mild impulses. This can be a glance of the player or a noise from a visible location. Entering this state will cause the AI to look over to the location of the event that instigated it.

Look

This task will stop the AI in its track and will make it look over at the suspicious event. This is set to a timed value, if nothing is seen to fully alert it in that time it will proceed with its 'default' state behaviour.

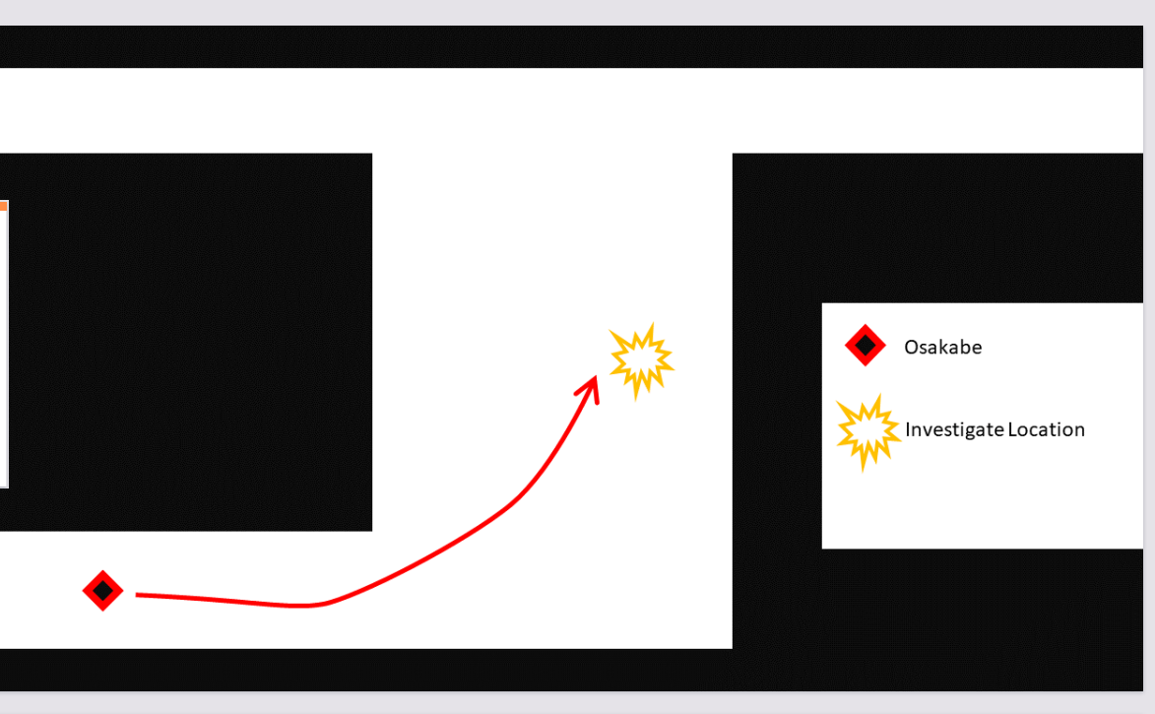


Investigate

Investigate is initiate when an impulse can't be directly located. For example when a noise from a room over is hearing but its origin can't be seen.

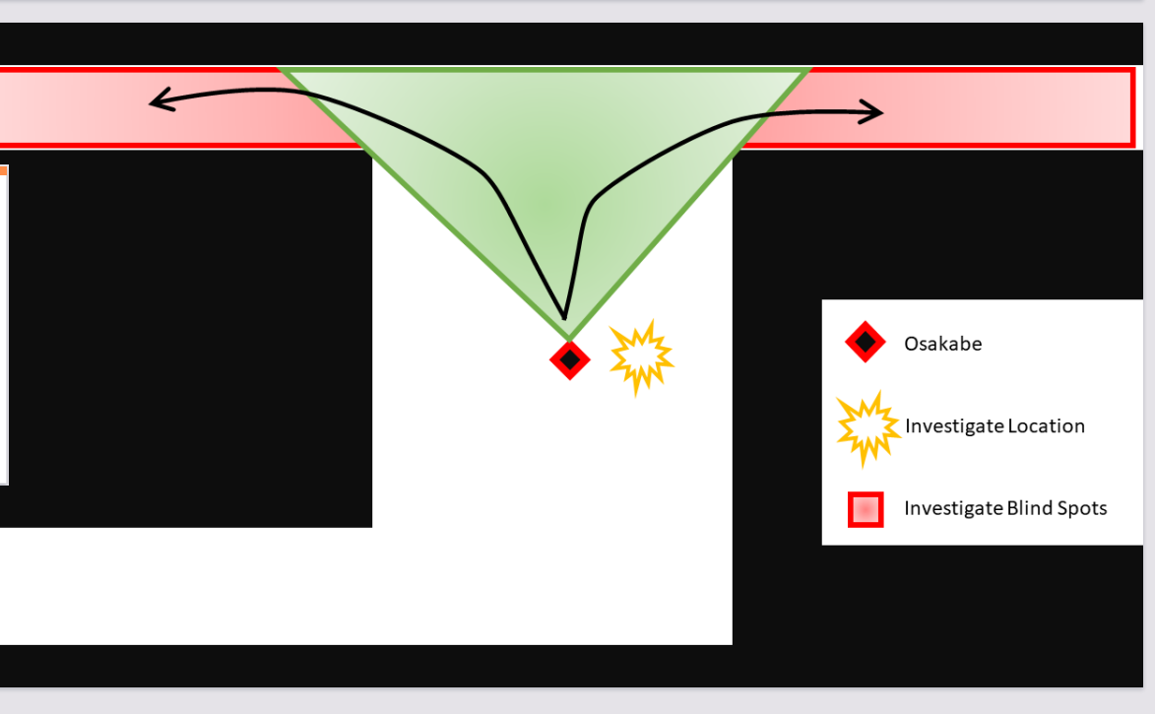
Inspect

This is its initial step. The AI will walk over to the area of origin. If this happens during patrolling or other general movement it will abort those as this [inspecting] is of a higher priority.



Search

Once the AI is in the area that instigated the investigation it will test what it can't see. After scoring them the AI will search one possible location where the player could be hiding. If the player isn't found the AI will return to its default state.

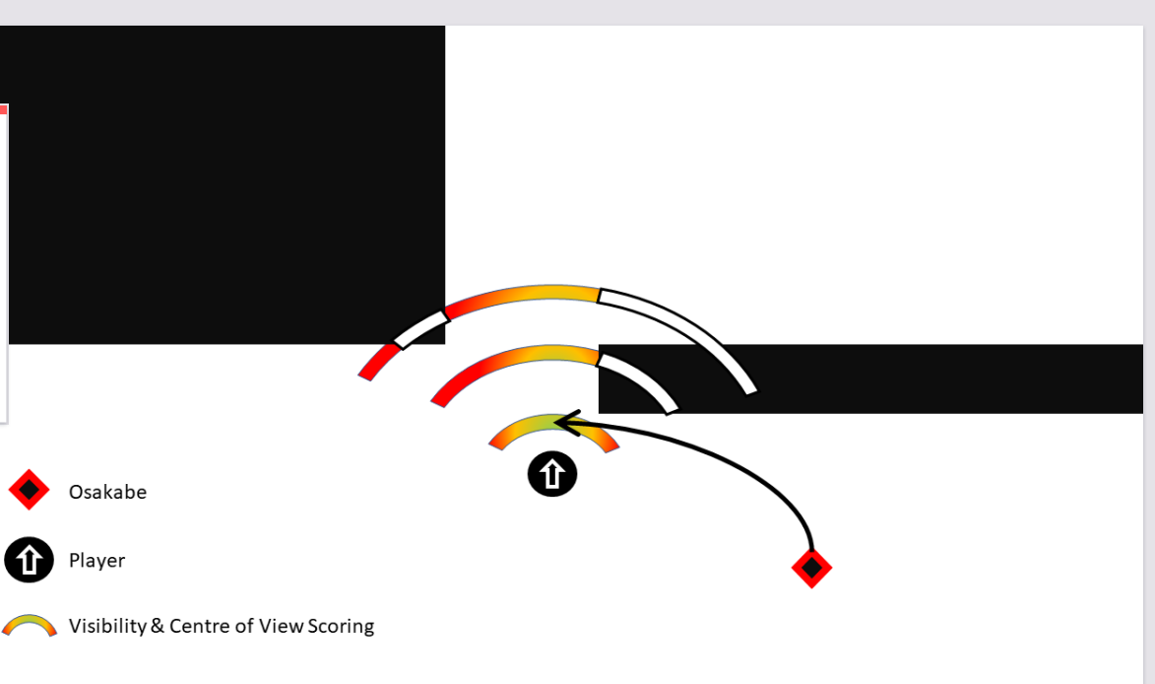


Alert

At this point the AI is fully aware of the player. This is a brief moment of intense, pants shitting gameplay.

Dashing

The AI will dash at incredible speeds (almost jump-scare like) to a location nearby the player and right in the center of their view. Which then forms the 'attack' dynamic together with the knowledge that looking at the AI deals the player damage.



Background

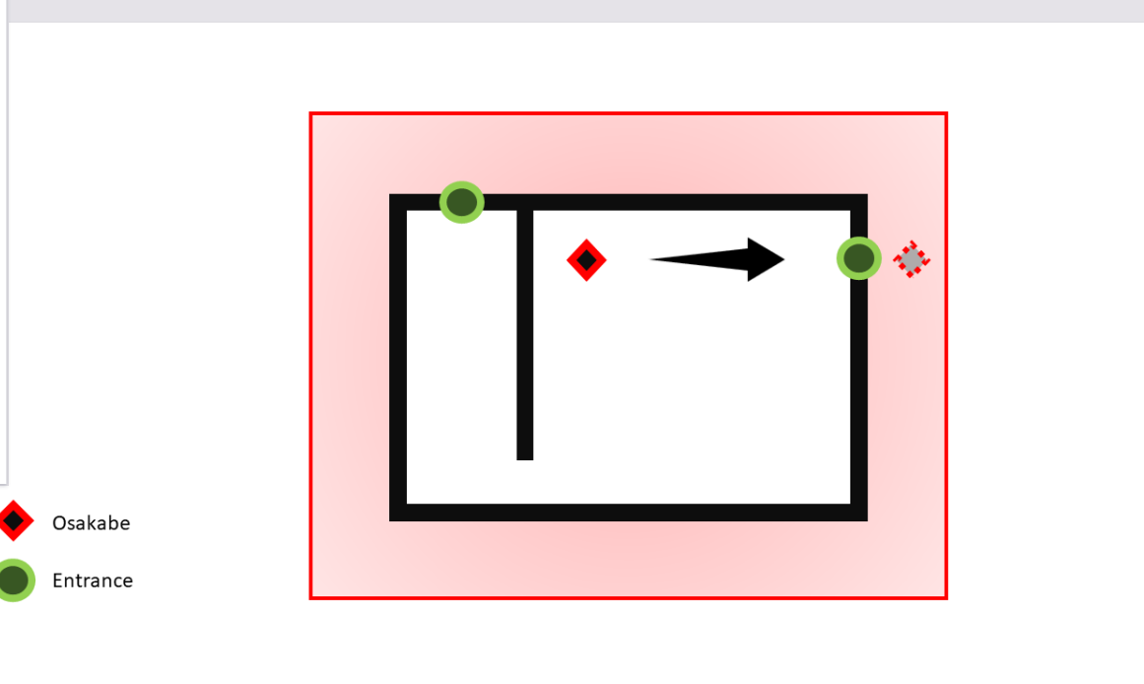
We use the background area to give the player some space to breath. This area is inaccessible to the player, meaning that is the AI is in it, it isn't an immediate threat to the player.

Secondly we use this to as a means for the AI to fast travel without using actual (unfair) teleportation. When in the background the AI can sprint in a shortcut, meaning they're able to get anywhere faster then the player yet it still takes time (making it fair and easier for the player to anticipate (because there are more defined rules to it)).

Entering Background

The gateways between the foreground and the background. When the AI is told to move to the background it will move to a point behind the entrance that's has the shortest walking distance. This is important as an entrance a floor higher may be close by but is at a longer walking distance. As well as one way entrances (such as a ledge that can be jumped down from but not up). It may be the closest but if the AI can't jump up there it still has to walk around and find another way.

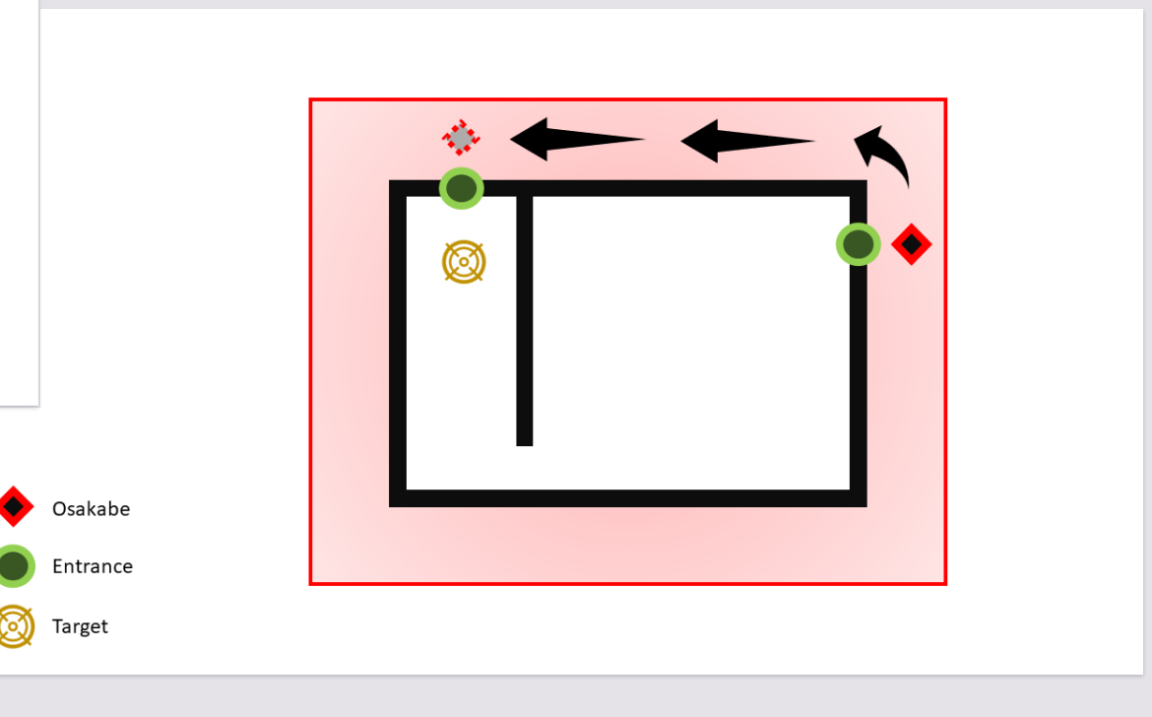
This task will abort all others as it's backing off, meaning it will ignore the player while it's 'running away'.



('In the Wind') Movement

Once in the background the AI will move to the entrance that's closest to its target location.

If at all possible it will do so without ever leaving the background area. If the specific entrance is inaccessible that way it will travel through the foreground. To make clear the AI is in it's background state it isn't visible but instead shows a smoky particle effect as feedback of it's movement to the player.



DETECTION

General Description

Senses through AIPerception

Sight

Hearing

Touch

Prediction

Knowledge

Knowledge Sense

Description

This 'sense' is basically gathering info from items in the world. In its simplest form it means the AI will inspect a door if it find the key that belongs to it is missing.

This can both be used to anticipate player actions as will as guide the player as the AI will inspect the door which relates to the key the player stole. This provides the player info on what the key is used for.

Future Plan

This 'sense' is not part of the mvp and is therefore not a must for the first iteration on the perception system. As it will need its own research and logic it will be put on the backlog for now.

RESEARCH

Detection System

Prediction Sense

Description

This sense is slightly more complicated as it isn't just a fact of seeing something yes or no but trying to combine info to anticipate the player's next move.

An over simplified example would be not seeing if the player left a room through exit one or two. If exit one is towards the player's objective then that's probably the way to go but if there's already a spirit animal over there the AI may decide to take exit two anyways to spread out and cover more ground.

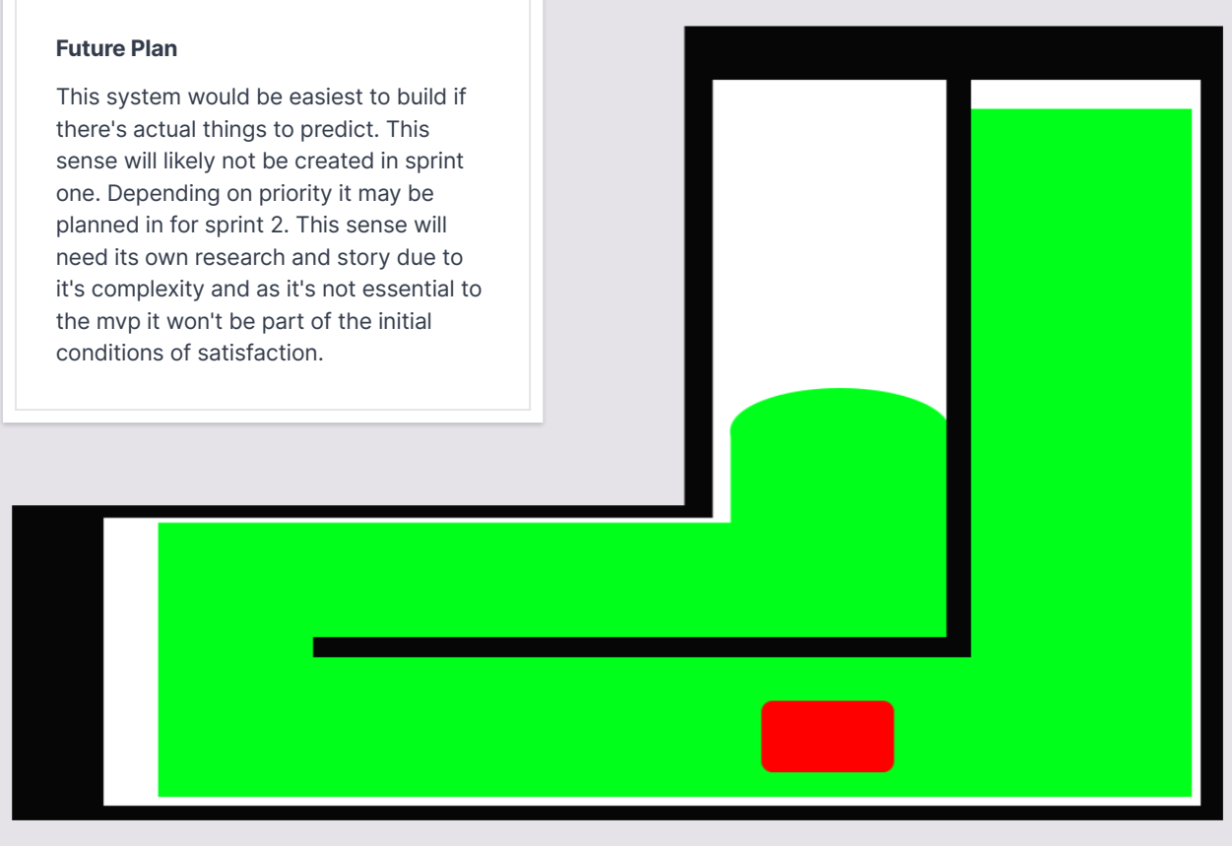
Future Plan

This system would be easiest to build if there's actual things to predict. This sense will likely not be created in sprint one. Depending on priority it may be planned in for sprint 2. This sense will need its own research and story due to it's complexity and as it's not essential to the mvp it won't be part of the initial conditions of satisfaction.

Touch Sense

Description

This is really just there to prevent the AI from looking stupid. This sense build off the assumption that the player at this immediate vicinity should've likely bee detected by sound or sight but there are edge cases where this might not be the case. For example is when both are in the incense smoke, and the player sits still, they would be undetectable. This sense will ensure that no matter what the conditions are, if the player and the AI are touching the player will always be detected.



Key; [RED]=listener; [GREEN]=hearing range; [BLACK]=walls;

Hearing Sense

Description

The hearing sense allow the AI to be perceptive to anything that is loud enough within a certain range. It allows for perceiving things behind walls or otherwise out of the view cone.

In contrast to sight this sense isn't always relevant. Where sight can always be seen, something can only be heard when it makes an actual sound. This means this sense will be triggered by player actions rather than their mere presence.

To prevent the experience from feeling unfair a perceived sound on it's own can never alert the AI. It can make it suspicious and make it decide to investigate. It can then confirm it's suspicion by detecting the player with their sight sense but the sound alone won't be enough to detect the player.

Sound doesn't always create tension. Especially reoccurring sounds you hear over and over again such as footsteps are easy to get used to. This tension can be created with some excellent feedback such as readable animations that indicate the AI heard you (1). You can thing of it like a deer eating to then quickly looking up to hearing something in the bushes.

Pathfinding

To mimic realistic sounds we'd have to account for sound traveling through walls. As actually simulating this is quite complicated we'll go by a simpler solution that seen in various games (2). Instead of using the distance to the sound we'll use the path finding distance. This way we'll be able to mimic sound traveling around walls rather than through them!

Referred Research Links;

(1): Blind Enemies (Link)

Other Sources;

(2): GDC AISummit: "Spaces in the Sandbox: Tactical Awareness in Open World Games" (Link)

Sight Senses

Description

Sight will be the AI's dominant sense. Where senses such as hearing and touch can be seen as only relevant when triggered the sight will always be relevant even when nothing is perceived. What this specifically means is that when the AI wants to inspect something it will do so by looking and not by trying to listen for it or try to touch it. Secondly it means that only the sight sense can actually detect something. If something is heard, it first needs to be confirmed by sight before the AI will act upon it.

For the actual sight senses and view cones we primarily looked at the AI from Alien: Isolation (1). The reasons for this is their multi view cone set up. While this set up may sound like over complicating it, it is actually very helpful for AI that roams around the world freely. In contrast to set patrol paths we as designers have less direct control over where the AI looks as this isn't scripted in. By using multiple view cones we extent what the AI can see without it being unfair by giving it eyes on the back of its head (2). Read further below to read about each sight sense's unique purpose and functionality.

Default

This is the basic most version when it comes to view cones. It's the longest and will be roughly 75 degrees wide (believable angle for normal human vision). This sense will be active on default and out of all the individual senses this one sits at the very core.

A player entering this cone will not be immediately detected but will have to be seen for a certain amount of time to allow for a more fuzzy detection (preventing unfair experiences of the AI seeming overly perceptive). The rate at which this goes will have several modifiers to it such as the distance to the target. These modifiers should be easily scalable and be applied to for example the player being less noticeable when crouched. Although some games make it a very front stage mechanic (3) we'll keep it limited to hiding spots in the level to prevent the player from becoming/feeling too powerful.

Peripheral

This sense works parallel to the default sight sense. It's purpose is to detect at a very wide angle (around 120 degrees) but at a very short range to mimic peripheral vision. This pretty much ensures the AI isn't oblivious to things happening right next to it.

As this is at such a short range the distance modifier would be a lot less as well as the overall detection rate being a lot lower. This is really just there to be able to see a player sprinting by at full speed.

To ensure a fair experience this view cone will not actually detect the player. Instead it will become 'suspicious' and the perception would have to be confirmed by one of the other sight senses. This way the AI has to look at the player before detecting them. This makes it very clear to the player when they're being seen.

Focus

This sight sense as to be activated upon which it will enable itself and disable the other sight senses.

With focus vision it's detection rate will be a lot faster as it was triggered by a noise for example.

Its cone is slightly shorter than the default but a lot tighter, to mimic tunnel vision. Al though this is a threat at first glance it can also be used to the player's advantage by creating a distraction that will make the AI focus on that allowing the player a window to sneak by.

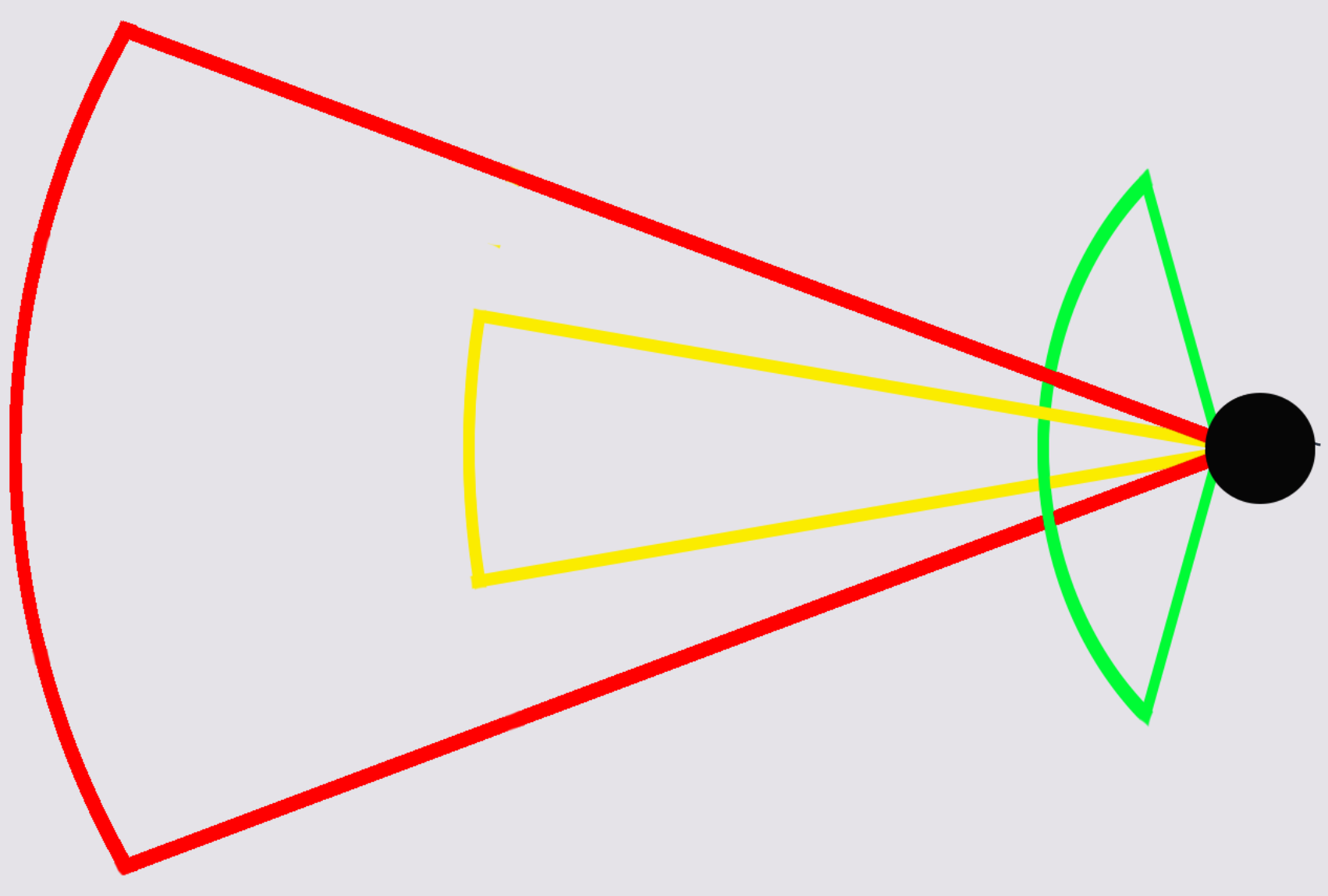
Referred Research Links;

(1): Alien Isolation Vision Sense (Link)

(3): Conditional Stealth (Link)

Other Sources;

(2): AI and Games: "Revisiting the AI from Alien: Isolation" (Link)



Key; [RED]=default; [GREEN]=peripheral; [YELLOW]=focus;

MOVEMENT

What To Expect Here

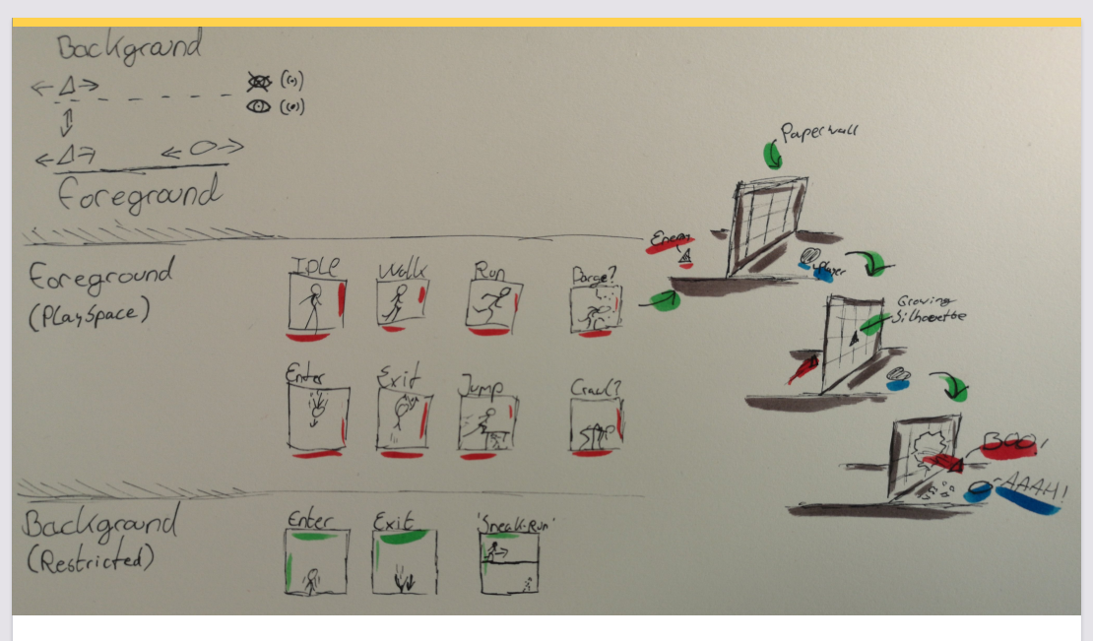
The movement for a large part is based on systems in the Unreal Engine. We make use of objects such as the Nav-Mesh and the Character Movement Component as the foundation of the AI's movement. Although we also use its pathfinding, our application is slightly different. However, in here we describe the movement itself, how this is used by the AI can be seen in the more interesting 'behaviour' section.

The main things covered here will be the components added to the existing Unreal systems. You can expect 'Rulebooks' and the 'Background' level layer as the main topics.

High-Level Description

The AI can move around the world at separate 'gaits'. For this we use a 'rulebook' system to update variables in runtime according to predetermined values. This means the AI never updates just one value (such as increase max speed) but updates all values in the 'rulebook' (so running isn't just an increase in speed but also in acceleration and rotation rate).

Secondly we make use of a 'background' layer. This is a space in the level only the AI can enter. We use these to control the pace, either by moving the AI there to give the player some space to breath or to give the AI a shortcut/secret passage to get anywhere in the level faster (without just teleporting it).



Movement Design Sketch, (Note that actions actions such as Barge and Crawl aren't part of the 'mvp')

RESEARCH

Enemy Movement System

CAUTION!

Some aspect of movement systems in Unreal that normally can be left to assumption are slightly different here. Example are having multiple nav-meshes and using obstacle areas as data for other systems. Please read the technical documentation to be sure you're clear on how some systems are used differently then one might expect.

BASIC MOVEMENT

'Rulebooks'

The Osakabe has four basic movement options. These are; walk, run, sprint and dash. These are set in runtime by using the 'Update Rulebook' macro and selecting the desired 'gait' on its input pin. These 'Rulebooks' are a set of data-only blueprints that store values of a struct of data. These values are then set by the AI as its own. This makes it easier to consistently change these in runtime as it will set all the variables and doesn't need to set each variables manually.

Walk

This 'gait' is used by the AI when there isn't much haste by its actions. Typically used with the patrol tasks (Patrol Move To and Patrol Area).

It's not particularly fast but it's search through the area and doesn't have a need to speed up.

This state also comes with slower acceleration and slower rotation rate and have no applied braking force. This means it will smoothly slow down as it approaches its destination.

Run

The run is typically used in moments where we need to get the AI somewhere before the moment has passed. This can be running towards something to investigate as we want the AI to search there before the player had plenty of time to get away. As well as when the movement destination is longer. If the AI would walk, by the time it reaches the target it is nowhere near where it actually needs to be, thus we run.

Compared to walking, all variables are slightly raised except for the braking force. This remains to be not applied.

Sprint

This 'gait' is unnaturally fast. It's only used in cases of the AI not being seen by the player. This can for example be when it's in the background. The background paths need to be faster for the AI so increasing its speed will ensure they're faster, even if the distance is longer.

As not all background areas are connected it can occur that the AI has to pass through the play space to get there. In this case the AI is invisible and only noticed by smoke-ish particles passing by. (note this is defined by the behaviour, this is not bound to this 'gait').

As this gait isn't seen by the player a lot of 'inevitability' settings can be removed. Instant acceleration, rotation rate and brake force are used here. In combination with a high max speed.

Dash

The dash is exclusively used during the alert state to get in the player's view. It's pretty much a jump scare that is just one step down from plain teleporting.

The reason it's this fast is because it needs to compete with the player's rotation rate. If the player looks away, the AI needs to get in center view just after.

As the distances are often short the AI doesn't use any acceleration. If it did it would never reach the desired velocity. The same goes with rotation, at those speeds it doesn't make sense for it to have any kind of turning circle, so it doesn't.

'BACKGROUND' MOVEMENT

Background

We use the background area to give the player some space to breath. This area is inaccessible to the player, meaning that is the AI is in it, it isn't an immediate threat to the player.

Secondly we use this to as a means for the AI to fast travel without using actual (unfair) teleportation. When in the background the AI can sprint in a shortcut, meaning they're able to get anywhere faster then the player yet it still takes time (making it fair and easier for the player to anticipate (because there are more defined rules to it)).

Areas

The background layer in the level is marked with a purposed actor. These don't need to be accurately placed but what is important is that the entire background area is overlapped and none of the playspace is.

Their only purpose is to be used in checks to see where the AI is. This information is then used in the behaviour tree.

Entrances

The 'entrances' are the gateways between the playspace and the background.

These have a front face and a background face. They come with a Nav-link to connect the navmeshes. These are set up for regular movement on a floor but a level designer can move the nav-link pods in the instance to set it up for vertecal movement i.e. jumping down. In those cases it also need to specified that the entrance can only be used in one specific direction. To make the AI aware it can jump down but not up.

Navigation

Every passage between the two layers are marked as 'obstacle' nav areas. Using a custom nav filter 'OsakabeDefault' when can use that to identify point in a different level layer as the AI. The cost of the obstacle is set to 10000, meaning we can use it to identify areas when we're aware that everything <10000 is in the same layer as the AI. Using the nav filter 'nullobstacle' we can filter out all the passages which would force the AI to move while staying in the level layer it is in. Please view the technical documentation for a more detailed view on the specific uses of the navigation system.

DIRECTOR AI

The Osakabe gets some info from a director AI. For example, when it's too far away for too long it will get push in the right direction.

Currently the Director is linked up however this was done in anticipation for block D.

Expect this section to get updated once it's being worked on in the future!