Dungeon Static Obstacles

28/03/2021

Players in order to collect objects, needed to survive, have to scavenge and raid parts of the city in a modular generated dungeon.
The **challenge** in those levels is provided by the different city security systems, the following threats are the Static Obstacles ones.

1. Cameras Security System

1.1 Cameras Characteristics

Security Cameras are objects that can be **placed** on **vertical surfaces** around the dungeon.
The only movement that SCs are allowed to perform is a **rotation on the vertical axis**.
The Fov of cameras is **always projected on the ground**, regardless of the height at which the physical camera is set and is of gold/yellow color.
When the character is caught by the camera's fov, the fov becomes red.

The movement is described with these variables:

- **(endAngleRotationValue):** final value of transform rotation value for the camera movement pattern (Vector3 variable).
- **(initialDeltaTime)**: time amount to complete the rotation to the starting rotation point to the final one (float variable).
- **(finalDeltaTime)**: time amount to complete the rotation to the final rotation point to the starting one (float variable).

There are other characteristic that can be influenced by other StaticObstacles:

- SCs can be active or not (**On/Off**), when **Off** they're **not rotating** and **won't** increase the **alert** value when players enter their FoV.
- They can be turned on when the players activate a **Tripwire** trap.

1.2 Interaction with Players

Security Cameras presents the following specifics in order to interact with players in the dungeon raids:

- **Field of View**:
  Is the element that defines the presence or not of the players into the SCs range view.
  FoV has to be implemented via a **spherecast system**, so it has to be regulated by the following values:
    - (**sphereDimension**): dimension of the sphere collider for the SCs (float variable).
    - (**angleFoV**): the field of view of cameras have the shape of a isosceles trapezoid, with the Base1 being always near the starting point of the Fov

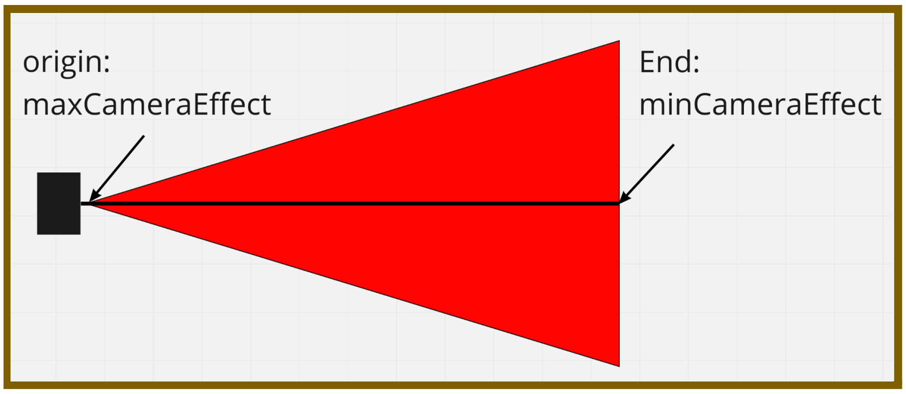
1.3 Alert Bar and Alert levels

- Alert it's a **float** value that goes **from 0 to 100**, and indicates how much the security system is aware of the player.
  - Increases based on how much time the character is spotted in the trigger of the cameras in **units per second** (**<u>alertIncrease</u>**).
  - It has discrete levels (**<u>alertLevel)</u>** that when reached activate the consequent Alert value, **in percentage** (**<u>alert</u>**).

Alert Level	0	10	20	30	40	50	60	70	80	90	100
Alert	100 %	105 %	110 %	115 %	120 %	125 %	130 %	135 %	140 %	145 %	150 %

  - Once players reach the maximum level of alert (100), the alarm is triggered and all enemies in the zone will run towards him.
  - In **co-op** mode the alert bar is **shared** between players.
  - Alert bar is **resetted** every time a **new** dungeon is started.
  - **alert increases the agent's trigger and field of view.**

- **Alert increases**:
  - When the character touches the fov it immediately gets its alert level increased, successive ticks of effect are received every X amount of time if the character stays in the Fov (**<u>secuirtyCameraRate</u>**).
  - The amount of how much the alert increases depends on how much **distance** from the Fov **origin** the character is when the tick of effect is dealt.
  - This value **linearly decreases** from a maximum at the Fov origin **(<u>maxCameraEffect</u>**) to a minimum at the very end of the Fov (**<u>minCameraEffect</u>**).



2. Force Field Barriers

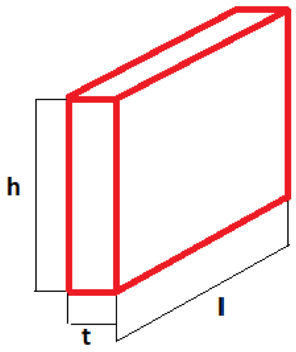
The fields are energetic **walls** which prevent players from passing on the other side, these walls are obstacles that lock some looting areas of interest.

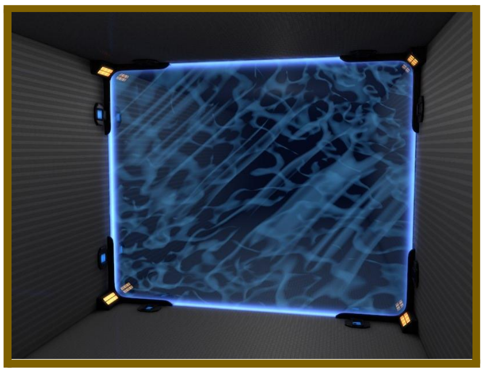
2.1 Barriers Characteristic

FFB are placed between buildings in the dungeon.
The dimension of FFB can change due to level design needs, even though it can mostly change in Length and Height to fit the gap between buildings.

There main characteristic of the FFB are:

- Present a collider, players can't walk through them (same for IA agents).
- If players collide with them there won't be any malus effects on them.
- Can be active or not (**On/Off**), when **Off** the respective GameObject is not active in the scene.



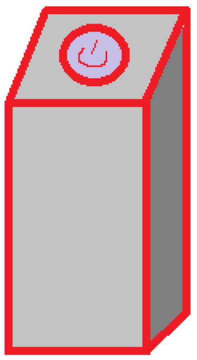


2.2 Single Switch FFB

FFB can be turned **Off** by players interacting with **switches** in the game level.

- FFB can be turned Off by a single or multiple Switches.
- Multiple FFB can be turned Off by a single Switch.

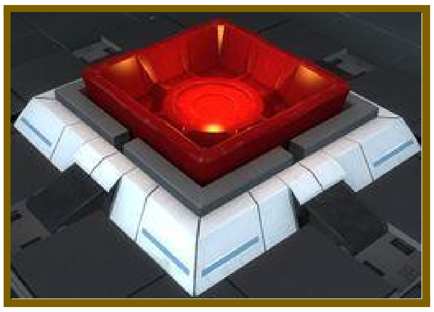
A Switch is an interactable object placed in the game level which can be turned Off.



2.3 Double Pressure Plates FFB

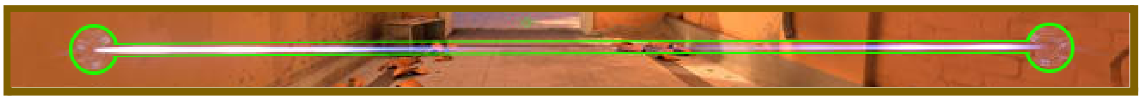
Players in order to turn off these FFB have to stay, at the same time, into two different **PressurePlateAreas.**
Once both players are into the PressurePlateArea the FFB turns Off until the end of the dungeon.

Note: Plates have to be represented with a specific sign in order to let the player know the correct couples of plates.



3. Tripwires

Tripwires are laser lines positioned between two vertical surfaces.
They are also placed at a defined height, players have the possibility to jump above them in order to avoid their effect.



There are two different types of Tripwire lines:

3.1 Activation Security Tripwire

- This Tripwire can only be triggered by players, when they collide with it.
- Once triggered, turns On one or more SCs or ST (Static Turrets).
- Once triggered, the Tripwire turns Off itself.
- A Tripwire triggered, once turned off, it can't be activated again.*

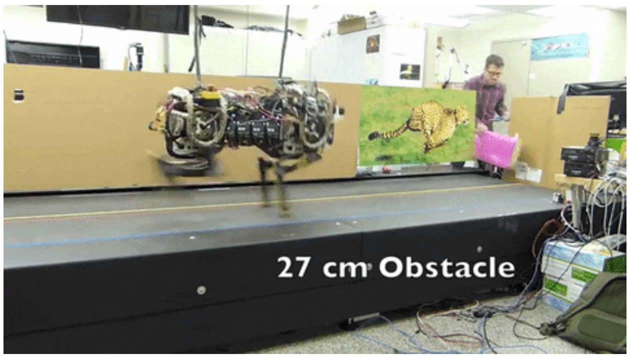
3.2 Damaging Tripwire

- This Tripwire can be triggered by both players and AI agents.
- This Tripwire can't be turned Off.
- If a Player triggers the Tripwire a defined damage to the Lifetime is dealt immediately, and if the player keeps staying in contact with the Tripwire the damage is dealt again every **X time**. (**tripwireLifetimeDamage)*IFDamageMod**
- If an AI agent triggers the Tripwire, the agent is damaged.

Both type of tripwire can move vertically and horizontally, those are the variables that manage this movement:

- **startPoint** (Vector 3, private), indicates the starting point of movement.
- **movementRange** (Vector 3), indicates the movement ending point of.
- **waitAfterMove** (float), indicates the amount of time a tripwire can stay still after reaching the ending/starting point.
- **duration** (float), indicates the duration required for the tripwire to reach the final point of its movement.

Note: these two types of Tripwires have to be different in the color layout for the players.



4. Static Turrets

Turrets are objects that can shoot projectiles to characters, dealing **damage**

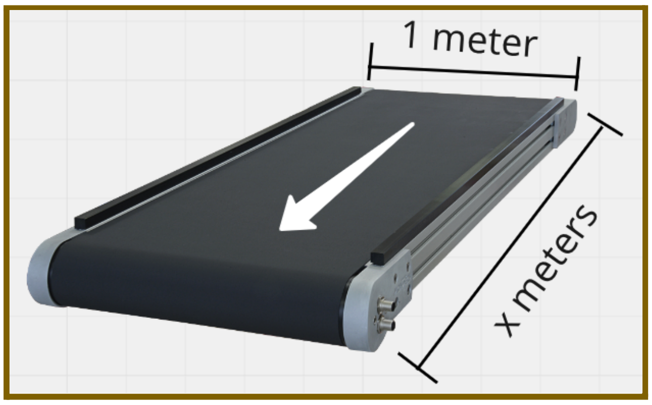
- Turrets can be **Activated or Deactivated,** by default or triggered by the Activation security tripwires.
- turrets have a trigger with an **x radius, (turretRange),** if the trigger of the turret overlaps with the trigger of the character (**aggroRange)**, the turret can spot the character, and if **Activated** aim at his collider, and shoot at him.
- turrets have an **x speed** at which they can rotate to aim at the character (**aimSpeed**)
- turrets shoot projectiles that deal an **x damage (turretDamage)** to the character, move at an **x speed** (**turretBulletSpeed**) and have an **x time** between shots (**turretRatio**)
- the turret range is modified by the alert level; **turretRange x alert x if AlertMod**

Note: the following paragraphs are not implemented in the final version of the game.

5. Conveyor Belt

Conveyor belts are types of surfaces that influence the movement of the character that walks on them.

- A conveyor belt has a **width of 1 meter** and a variable length in meter (**conveyorLength**)
- The conveyor belt has a direction in which it moves at an **x speed in ms** (**conveyorSpeed**)
- The conveyorBelt makes the character move **faster** by (**conveyorSpeed**) if the character is moving **along** the conveyor direction: [(**walkingSpeed**)*(**walkingSpeedMod**)] **+ (conveyorSpeed)**
- The conveyorBelt makes the character move **slower** by (**conveyorSpeed**) if the character is moving **contrary** the conveyor direction [(**walkingSpeed**)*(**walkingSpeedMod**)] **- (conveyorSpeed)**
- If the character **stands still** on the conveyor belt, it moves along the **direction** the belts is moving at the pace of **(conveyorSpeed)**
- conveyor belts does not have effect on flying agents
- conveyor belts have effect on terrain agents
- multiple conveyor belts can be stacked side by side



6. Flashing light

Flashing lights are lights that are positioned inside a game object (lamps, light pole, ecc..) that can alternate between different states following a pattern.

- flashing lights can be spot lights or point lights
- flashing lights have the classic light parameters (radius, intensity ecc.)
- flashing lights have 3 states: **On**, **flashing**, **Off**
- in the **On** state the light is on for an x amount of time in s
- in the **flashing** state the light is switch between on and off at a fixed speed for an x amount of time in
- in the **Off** state the light is off for an x amount of time in s
- The pattern of a light can be implemented as a **list** of states, with every state having a time of duration in **seconds** before moving to the next one, by setting zero on a state, that state can be **skipped**.

Exemple:
On :3
Flashing :2
Off :0
On :4
Flashing :2
Off : 0
On : 0
Flashing :3
Off : 4