Documentation

Grid-based movement controller

By Evengy

Contents

Description.

The purpose of this tool is to provide a head start for the development of grid-based dungeon crawler games.

The "Grid-based movement controller" provides the basic functionality of a first-person grid-based movement system. It features a grid system based on a responsive grid unit called a "Tile." The grid system allows for decoupling of tile mechanics from its model, if necessary, and the ability to design the level art style separately from the grid. Each tile on the grid can be dynamically switched as walkable or not, depending on the level design.

Key features.

This tool can be used for developing first-person grid-based dungeon crawlers and similar games.

A sample level demonstrates the implementation of various elements, such as:

- Stairs
- Elevators and moving platforms
- Ladder
- Tunnel (crouching mechanics)
- Dynamic obstacle (door)
- Hidden path

All these elements utilize the grid system, which separates the design of player movement from the logic of the level elements. The player's movement mechanics rely solely on the grid system and are not dependent on the specific level elements. This allows for the implementation of additional elements without needing to modify the player's movement logic.

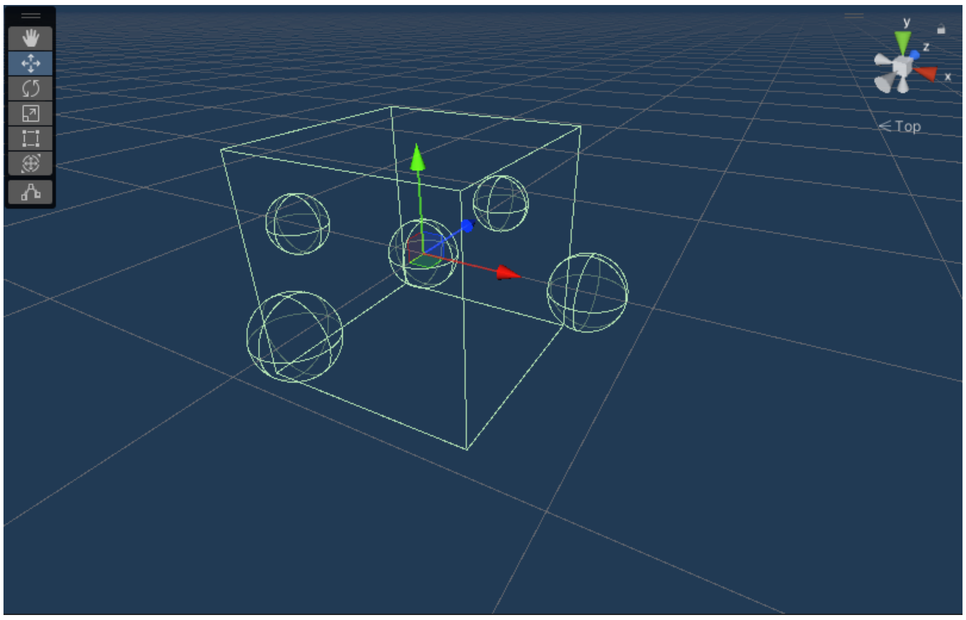
This is not an exhaustive list, as other level elements can also be implemented using this approach.

Grid-based movement system.

The core element of the grid-based system is a grid unit called a "Tile." Each Tile is a game object with the tag "Tile" and includes the following components:

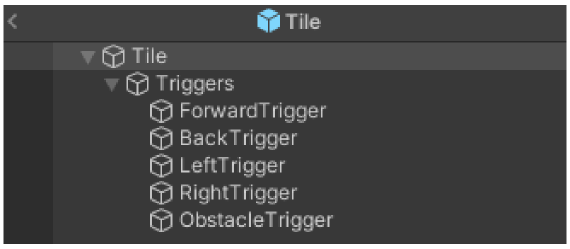
- Box Collider Component: The Tile has a box collider component, which provides the physical boundaries for the Tile object.
- Triggers: There are four triggers in each direction (forward, back, left, and right) that detect neighbouring tiles on the grid. These triggers facilitate movement and interaction between adjacent Tiles.
- Central Trigger: In addition to the directional triggers, there is a central trigger that detects obstacles within the Tile. This allows for obstacle detection and handling within the grid system.

By utilizing these components, the Tile serves as the fundamental building block for the grid-based system, enabling movement and interaction between tiles and handling obstacles within the grid.



Pic 1 – Tile prefab view.

Each Tile object has a child object called "Triggers" that includes four directional triggers and one obstacle trigger:



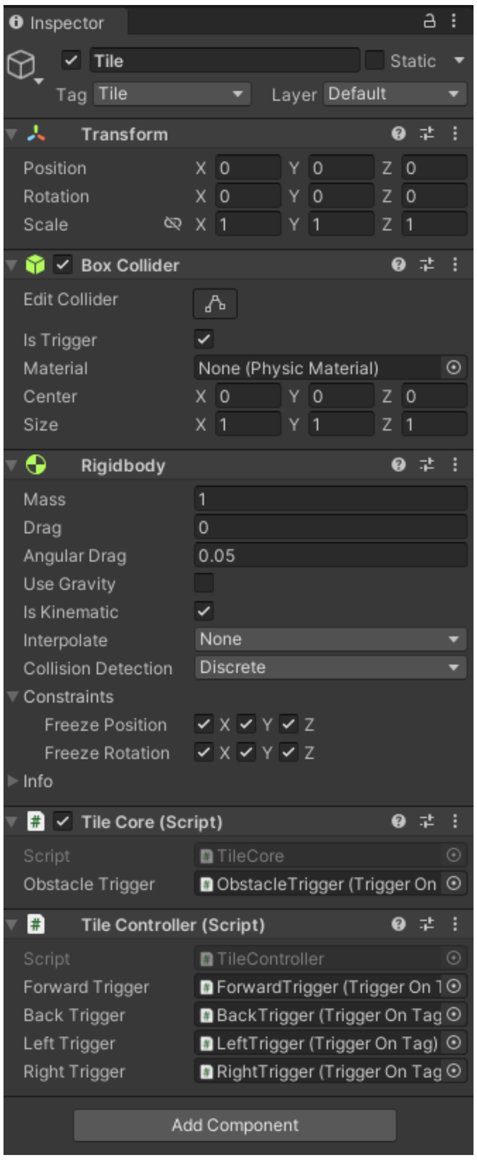
```
Tile
▼ Tile
  ▼ Triggers
      ForwardTrigger
      BackTrigger
      LeftTrigger
      RightTrigger
      ObstacleTrigger
```

Pic 2 – Tile object hierarchy.

A model of the floor could be added as a child object of the Tile, if necessary.

Each Tile object has the following components:

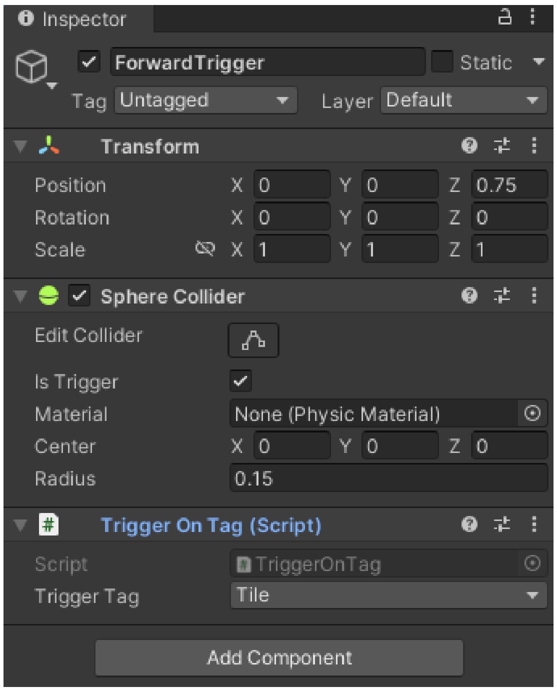
- Box Collider Component: This component has the "Is Trigger" checkbox enabled, and it is required to define the scope of the Tile.
- Rigidbody Component: This component has the "Is Kinematic" checkbox enabled, and all Constraints checkboxes are also enabled. It is required to detect collider trigger events.
- TileCore and TileController Components: These components are required to handle navigation on the grid. For more details, please refer to the class reference.



```
Inspector
✓ Tile                                    Static
Tag Tile                  Layer Default

Transform
Position          X 0    Y 0    Z 0
Rotation          X 0    Y 0    Z 0
Scale             X 1    Y 1    Z 1

✓ Box Collider
Edit Collider
Is Trigger          ✓
Material            None (Physic Material)
Center            X 0    Y 0    Z 0
Size              X 1    Y 1    Z 1

Rigidbody
Mass                1
Drag                0
Angular Drag        0.05
Use Gravity
Is Kinematic        ✓
Interpolate         None
Collision Detection Discrete
▼ Constraints
  Freeze Position   ✓ X ✓ Y ✓ Z
  Freeze Rotation   ✓ X ✓ Y ✓ Z
► Info

✓ Tile Core (Script)
Script              TileCore
Obstacle Trigger    ObstacleTrigger (Trigger On

Tile Controller (Script)
Script              TileController
Forward Trigger     ForwardTrigger (Trigger On 1
Back Trigger        BackTrigger (Trigger On Tag
Left Trigger        LeftTrigger (Trigger On Tag)
Right Trigger       RightTrigger (Trigger On Tag

Add Component
```

Pic 3 – Tile object components.

Each of the four directional triggers of the Tile object has a TriggerOnTag component with a Trigger Tag set to "Tile" and a Sphere Collider component with the “Is Trigger” checkbox enabled.



Inspector

ForwardTrigger ✓ Static
Tag Untagged Layer Default

Transform
Position X 0 Y 0 Z 0.75
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

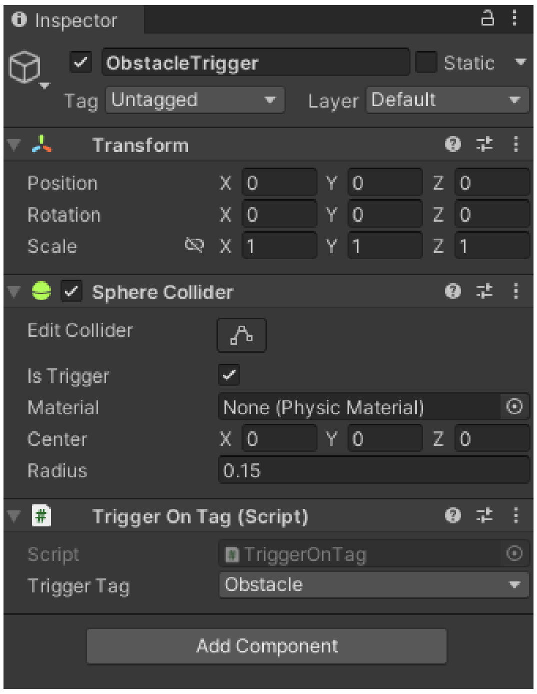
Sphere Collider ✓
Edit Collider
Is Trigger ✓
Material None (Physic Material)
Center X 0 Y 0 Z 0
Radius 0.15

Trigger On Tag (Script)
Script TriggerOnTag
Trigger Tag Tile

Add Component

Pic 4 – The Forward Trigger component (similar for the other directional triggers with varying sphere collider position values)

The Obstacle Trigger of the Tile object has the Trigger Tag set to "Obstacle". For more details on the TriggerOnTag component, please refer to the class reference.



Inspector

ObstacleTrigger ✓ Static
Tag Untagged Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Sphere Collider ✓
Edit Collider
Is Trigger ✓
Material None (Physic Material)
Center X 0 Y 0 Z 0
Radius 0.15

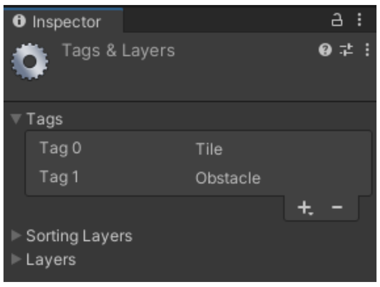
Trigger On Tag (Script)
Script TriggerOnTag
Trigger Tag Obstacle

Add Component

Pic 5 – Obstacle Trigger components.

Scene set up.

1) Create Tags named “Tile” and “Obstacle”:



Inspector

Tags & Layers

Tags

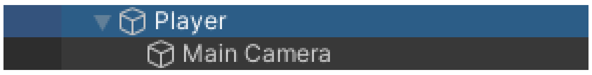
Tag 0	Tile
Tag 1	Obstacle

Sorting Layers

Layers

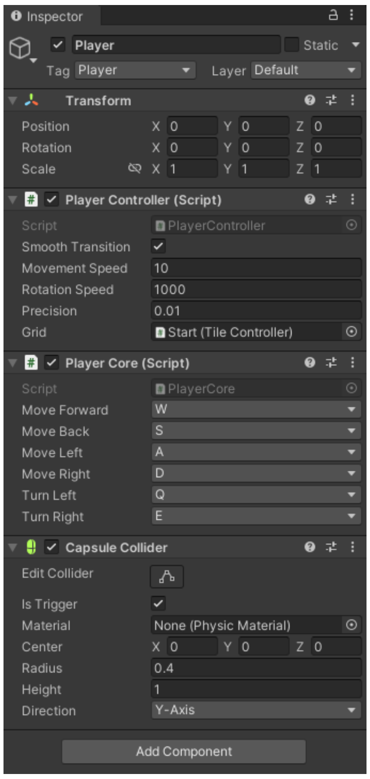
Pic 6 – Required additional tags.

2) Set up a level by either using the Tile prefab from the sample scene or creating a new Tile with the components explained in the grid-based movement system section. Create various elements within the level.
3) Create a Player game object and make the Main Camera a child object of the Player game object.
4) Attach the PlayerCore, PlayerController, and Capsule Collider components to the Player game object. Set the starting tile of the grid to the "Grid" property of the PlayerController component. Adjust parameters within the PlayerController component and assign movement keys in the PlayerCore component:



Player

Main Camera



Inspector

Player ✓ Static

Tag Player Layer Default

Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Player Controller (Script) ✓

Script	PlayerController
Smooth Transition	✓
Movement Speed	10
Rotation Speed	1000
Precision	0.01
Grid	Start (Tile Controller)

Player Core (Script) ✓

Script	PlayerCore
Move Forward	W
Move Back	S
Move Left	A
Move Right	D
Turn Left	Q
Turn Right	E

Capsule Collider ✓

Edit Collider	
Is Trigger	✓
Material	None (Physic Material)
Center	X 0 Y 0 Z 0
Radius	0.4
Height	1
Direction	Y-Axis

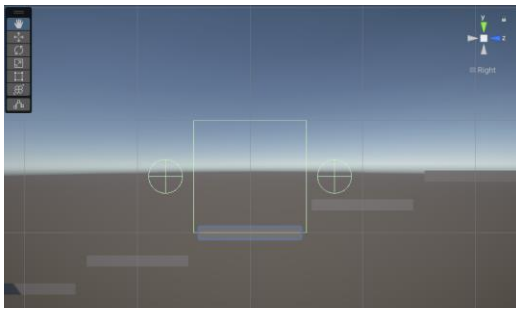
Add Component

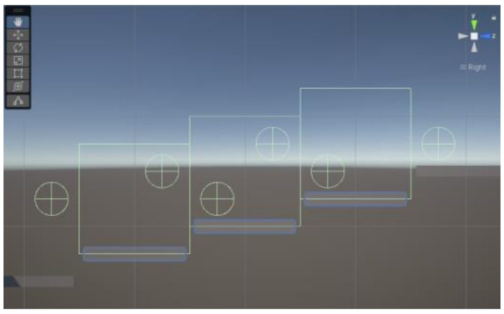
Pic 7 – Player game object’s components set up.

Level elements implementation.

- Stairs

Each tile is shifted along the Y-axis, but the trigger still intersects with the box collider of the next tile. As a result, the player controller will move towards the position of the next tile, which is elevated by a delta Y value relative to the current tile:

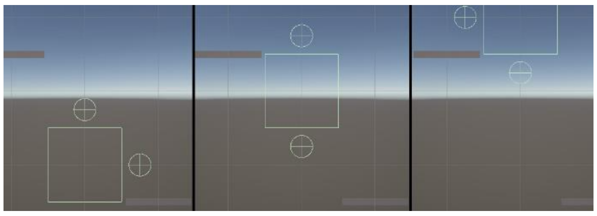

Pic 8 – Tile collider with its Forward and Back triggers.

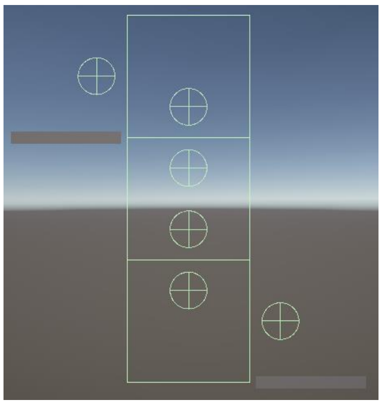

Pic 9 –Forward and Back triggers intersections with adjacent tile colliders.

- Ladder

Ladder in a sample level example requires adjusting the positions of the ForwardTrigger and BackTrigger.

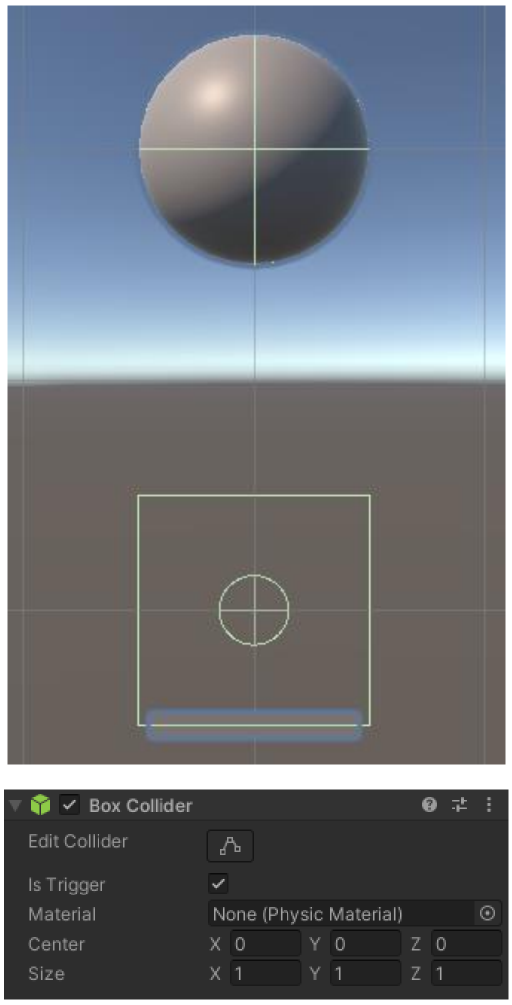
For each tile of the ladder, the positions of both the Forward and Back triggers should be adjusted to intersect with the next adjacent tile vertically, instead of horizontally like the rest of the tiles on the grid. This adjustment enables the player to move vertically when transitioning to the next tile.


Pic 10 – Ladder tiles: Tile colliders with Forward and Back triggers.


Pic 11 – Forward and Back triggers placement in the ladder and their intersections with adjacent vertical tiles.

- Obstacle

When an obstacle touches the obstacle trigger (sphere collider) of a tile, the main collider (box collider) of the tile disappears. This causes adjacent tiles to be unable to "see" this tile, resulting in their inability to provide the player with the position of the next tile. Consequently, the player cannot move towards the next tile.



Box Collider

Edit Collider

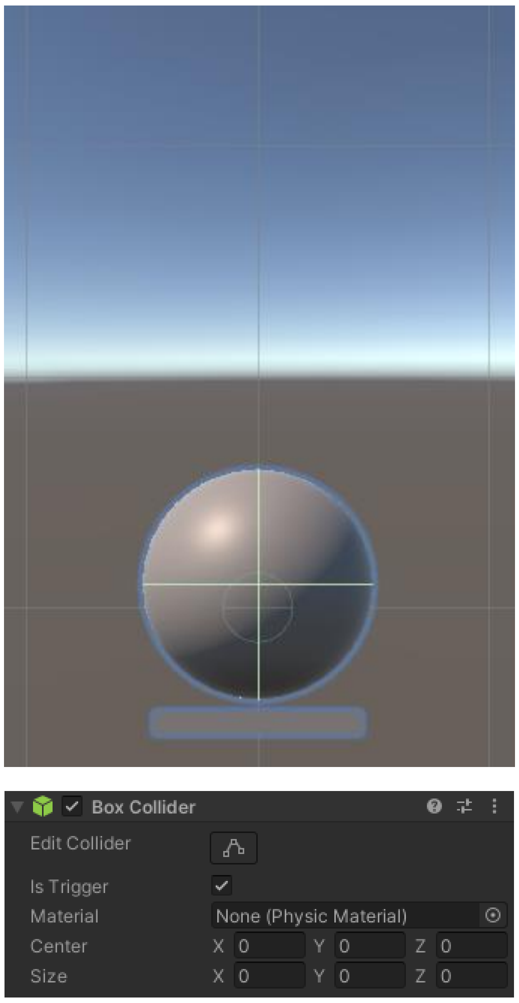
Is Trigger ✓

Material None (Physic Material)

Center X 0 Y 0 Z 0

Size X 1 Y 1 Z 1

Pic 12 – Obstacle and tile interaction. Tile is "walkable".



Box Collider

Edit Collider

Is Trigger ✓

Material None (Physic Material)

Center X 0 Y 0 Z 0

Size X 0 Y 0 Z 0

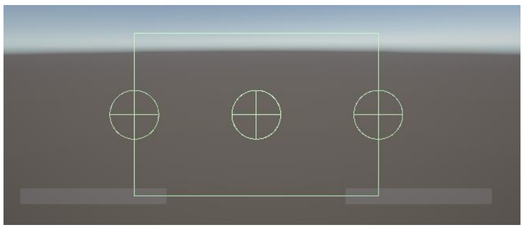
Pic 13 – Obstacle and tile interaction. Tile is not "walkable".

- Hidden Path

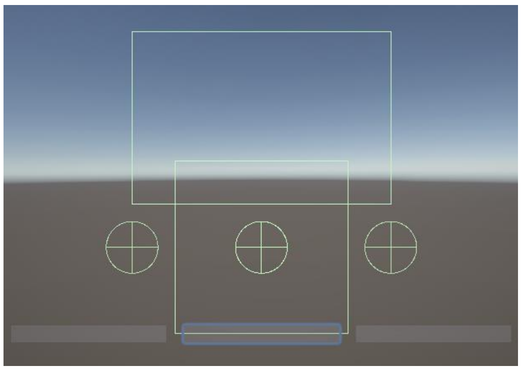
Obstacle mechanics can also be utilized for walls, doors, hidden rooms, or hidden paths that can be accessed under certain conditions (e.g., a player casting a spell to fly over water tiles). These mechanics provide versatility in level design.

In the sample scene, a hidden path is provided as an example of such a level element. When the player is next to the hidden path and presses the spacebar, it opens a path across the gap on the grid. For detailed information, please refer to the script reference.

The hidden path is a game object tagged as "Obstacle" with a Tile set as its child object. To open the path, the box collider of the hidden path object should be raised above the obstacle trigger of the tile, allowing the tile to become "walkable.":



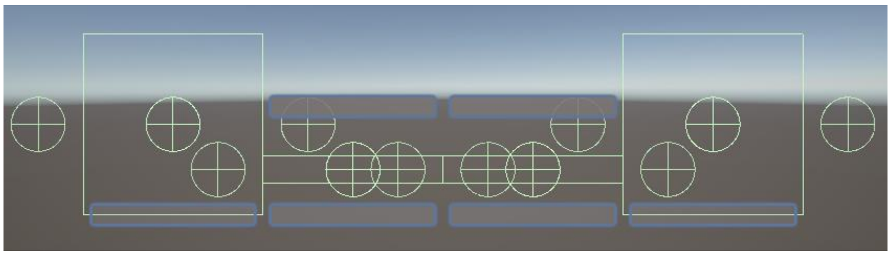
Pic 14 – Hidden Path. Tile "unwalkable".



Pic 15 – Hidden Path. Tile "walkable".

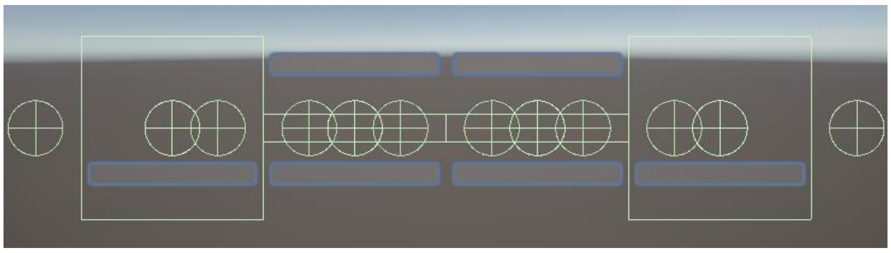
- Tunnel

The inner tiles of the tunnel have shorter main colliders, rendering them unwalkable for the outer tiles:



Pic 16 – Tunnel. Inner tiles "unwalkable" for the outer tiles.

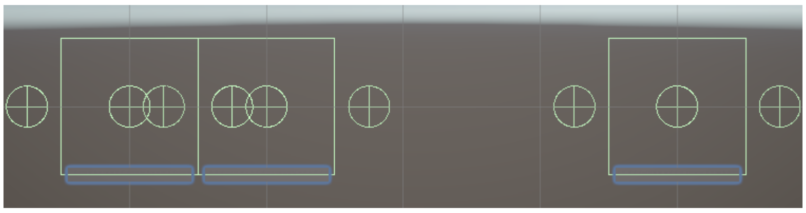
To traverse through the tunnel, the outer tile of the tunnel needs to descend so that its trigger can reach the collider of the first tile within the tunnel. This allows it to be recognized as the next walkable tile. Once the player crouches inside the tunnel, they will follow the tunnel's design based on the grid:



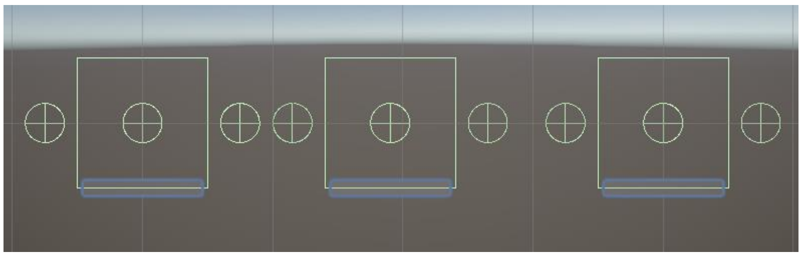
Pic 17 – Tunnel. Inner tiles “walkable” for the outer tiles.

- Moving platforms

Moving platforms utilize the Tile grid unit. The player can step on and off the platform only if they are standing on a grid tile that is in close proximity to the platform, allowing the player to identify it as an adjacent tile:



Pic 18 – Horizontal moving platform. The platform is “walkable” for the left Tile and “unwalkable” for the right Tile.



Pic 19 – Horizontal moving platform is “unwalkable” for both left and right Tiles.

A vertical moving platform can be designed in a similar manner. For an example of its implementation, please refer to the sample scene.

Script reference.

Navigation

Classes

TileCore

Description

This component defines a tile. Each Tile game object should have this component. It includes an obstacle collider that triggers when an "Obstacle" tag is detected. The tile is considered "unwalkable" when an obstacle is detected.

Properties

obstacleTrigger	refers to the TriggerOnTag component, which will have its "IsTriggered" property set to true if an object with the tag "Obstacle" intersects its collider.

Fields

tileCollider	main collider of the Tile object.
colliderDefaultSize	default size of the main collider.

Private Methods

Start	initializes the tileCollider and colliderDefaultSize.
Update	adjusts the size of the tileCollider based on the "IsTriggered" property of the obstacleTrigger.

TileController

Description

This component provides the functionality for grid-based movement. It includes four triggers that represent the four directions. When the player moves in a specified direction, it returns the adjacent tile object.

Properties

forwardTrigger	refers to the TriggerOnTag component of the “Forward” trigger object.
backTrigger	refers to the TriggerOnTag component of the “Back” trigger object.
leftTrigger	refers to the TriggerOnTag component of the “Left” trigger object.

rightTrigger	refers to the TriggerOnTag component of the “Right” trigger object.

Public Methods

GetTile	returns the adjacent tile based on the provided parameter "direction".

Private Methods

ForwardTile	returns the TileController component of the adjacent tile in the forward direction if the property "IsTriggered" of the forwardTrigger is true. Otherwise, it returns itself.
BackTile	returns the TileController component of the adjacent tile in the back direction if the property "IsTriggered" of the backTrigger is true. Otherwise, it returns itself.
LeftTile	returns the TileController component of the adjacent tile in the left direction if the property "IsTriggered" of the leftTrigger is true. Otherwise, it returns itself.
RightTile	returns the TileController component of the adjacent tile in the right direction if the property "IsTriggered" of the rightTrigger is true. Otherwise, it returns itself.

Enumerations

Direction

Description

keeps the directions with their corresponding Euler degrees values.

Properties

Forward	0 degrees
Left	270 degrees
Back	180 degrees
Right	90 degrees
Round	360 degrees

Player

Classes

PlayerCore

Description

Defines the movement keys and requires the PlayerController component.

Properties

moveForward	defines the key code for stepping forward
moveBack	defines the key code for stepping back
moveLeft	defines the key code for stepping forward left
moveRight	defines the key code for stepping forward right
turnLeft	defines the key code for turning left
turnRight	defines the key code for turning right

Private Methods

Start	initializes the PlayerController component.
Update	executes methods of the PlayerController component based on the pressed key.

PlayerController

Description

This component utilizes the grid to enable movement and rotation functionality.

Properties

smoothTransition	defines whether the player's movement and rotation are instant or not.
movementSpeed	defines the movement speed.
rotationSpeed	Defines the rotation speed.
precision	defines the precision with which the player should be placed on the Tile's position
grid	defines the start tile of the grid.
IsBusy	This property is set to true when the player is moving or rotating.

Fields

currentDirection	represents the direction the player's camera is facing and is defined by the enumeration "Direction".
targetTile	represents the next tile that the player should move towards.
TargetRotation	represents the rotation that the player should rotate towards.

Private Methods

Start	initializes the currentDirection field as "Forward", the targetTile field as the grid parameter, and the targetRotation based on the value of the currentDirection.
Update	executes the Move and Rotate methods if the IsBusy property is true, indicating that the targetTile has been assigned but not yet reached or the targetRotation has changed but not yet reached.
Move	moves the player towards the position of the targetTile.
Rotate	rotates the player towards the Euler angles of the targetRotation.
LocalDirection	returns the direction relative to the current direction based on the provided parameter. For example, if the currentDirection is "Left" and the parameter direction is "Left", it returns "Back".

Public Methods

MoveTowards	reassigns a new targetTile based on the provided "direction" parameter relative to the current direction.
RotateTowards	reassigns the currentDirection based on the provided "direction" parameter relative to the current direction.

Helpers

Classes

TriggerOnTag

Description

This helper component has a state "IsTriggered" when detecting a specified tag. All tags should be specified in a Tag enum. It is used by TileController to identify adjacent tiles, with the tag "Tile" specified for that purpose. It can also be used by other components to identify proximity with an object of a certain tag.

Properties

triggerTag	defines the tag that should set the state of the component to "IsTriggered".
IsTriggered	defines the state of the component. It is set to true if the gameobject's collider intersects with a collider of an object with the specified tag.
TriggerObject	refers to a game object with the specified tag that intersects with this component's game object.

Private Methods

OnTriggerEnter	sets "IsTrigger" to True if the component interacts with an object of the specified Tag, and saves the object as the TriggerObject property.
OnTriggerExit	sets "IsTrigger" to False if the component stops interacting with an object of the specified Tag, and assigns null to the TriggerObject property.

Enumerations

Tag

Description

This enumeration keeps the tags that can be referenced in the code.

Level

Classes

HiddenPathController

Description

hides or reveals the path by descending elevating the main collider over the obstacle collider of the tile.

Properties

maxElevation	represents the relative elevation value for the y-coordinate of the main collider.
revealPath	represents the key code that triggers the hiding/revealing of the path.
revealTileModel	refers to the model of the tile that will be revealed.

Private Methods

Start	initializes a TriggerOnTag component, the reveal collider (which is the main collider assigned to the object tagged as "Obstacle"), and the default position of the reveal collider.
Update	holds the functionality for hiding and revealing the path.

CrouchController

Description

The CrouchController component descends the position of the assigned Tile.

Properties

crouch	The KeyCode that triggers the Descent functionality
descension	The y value of the relative descension of the Tile's position.

Private Methods

Descent	Descends the game object's position by the given descension value provided in the component's properties.