Documentation

Grid-based movement controller

Contents

[Description. 2](#_Toc137558426)

[Key features. 3](#_Toc137558427)

[Grid-based movement system. 4](#_Toc137558428)

[Scene set up. 1](#_Toc137558429)

[Level elements implementation. 2](#_Toc137558430)

[Script reference. 3](#_Toc137558431)

# 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 of 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 elemnt of the grid-based system is a grid unit “Tile”. The tile is a game object with a tag “Tile”, 1 box collider component, 4 triggers in each direction that are detecting other tiles on the grid and 1 trigger in the centre to detect obstacles:

A picture containing screenshot, diagram, 3d modeling, graphics software

Description automatically generated

Pic 1- Tile prefab view.

Each Tile object has a child object Triggers that includes 4 directional triggers and 1 obstacle trigger:

A screenshot of a computer

Description automatically generated with medium confidence

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 these components:

* Box collider component with Is Trigger checkbox enabled. Required to define the scope of the Tile.
* A Rigidbody component with Is Kinematic checkbox enabled, all Constraints checkboxes enabled. Required to detect collider trigger events.
* TileCore and TileController components. Required to handle navigation on the grid. Refer to the class reference for details.

A screenshot of a computer

Description automatically generated

Pic 3 – Tile object components.

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

A screenshot of a computer

Description automatically generated

Pic 4 – Forward Trigger components (same for other directional triggers with the difference of sphere collider center values).

Obstacle Trigger of the Tile object has Trigger Tag set to “Obstacle”. Refer to the class reference for details on TriggerOnTag component.

A screenshot of a computer

Description automatically generated

Pic 5 – Obstacle Trigger components.

# Scene set up.

1. Create Tags “Tile” and “Obstacle”:

A screenshot of a computer

Description automatically generated with medium confidence

Pic 6 – Required additional tags.

1. Create a level with a grid using the Tile prefab from the sample scene or create a new Tile with components as explained in grid-based movement system part. Create elements within the level.
2. Create a Player game object, put Main Camera as a child object of the Player game object.
3. Add PlayerCore, PlayerController and a Capsule Collider component to the Player game object. Assign a starting tile of the grid to the PlayerController component’s property “Grid”. Adjust parameters in the PlayerController component and movement keys in the PLayerCore component:



A screenshot of a computer

Description automatically generated with medium confidence

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

# Level elements implementation.

* Stairs

Each tile is shifted by Y axis, but the trigger still intersects the next tile’s box collider. And a player controller will move towards the next tile’s position (elevated by a delta Y related to the current tile):

A picture containing screenshot, multimedia software, graphics software, software

Description automatically generated

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

A picture containing screenshot, text, multimedia software, graphics software

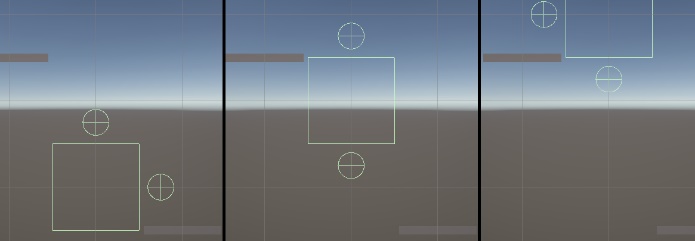
Description automatically generated

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

* Ladder

Ladder in a sample level example requires ForwardTrigger and BackTrigger’s positions to be adjusted.

Every Forward and Back trigger of each ladder’s tile should be adjusted to intersect next adjacent tile vertically instead of horizontal placement for the rest of the tiles on the grid. That allows player to move vertically when moving to the next tile.



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

A screenshot of a game

Description automatically generated with low confidence

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

* Obstacle

When obstacle touches the tile’s obstacle trigger (sphere collider), tile’s main collider (box collider) disappears. That causes adjacent tiles not to be able to “see” this tile. Therefore, they cannot provide player with the “next tile’s” position and player cannot move towards it.

A screenshot of a video game

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

Pic 12 – Obstacle and tile interaction. Tile is “walkable”.

A picture containing screenshot, sky, planet, space

Description automatically generated

A screenshot of a computer

Description automatically generated

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

* Hidden Path

Obstacle mechanics could be also used for walls, doors, hidden rooms, or hidden paths that could be entered on condition (for example a player could fly above the water tiles if casted a spell) and so on.

In the sample scene a hidden path is given as an example of such level element. And it will open a path across the gap on the grid when the player is next to it after pressing space. Refer to the script reference for details.

Hidden path is a game object tagged as “Obstacle” and a Tile set as a child object of the Hidden path object. To open the path a box collider of the hidden path object should be elevated above the obstacle trigger of the tile to make the tile “walkable”:

A screenshot of a video game

Description automatically generated with medium confidence

Pic 14 – Hidden Path. Tile “unwalkable”.

A screenshot of a computer

Description automatically generated with low confidence

Pic 15 – Hidden Path. Tile “walkable”.

* Tunnel

Tunnel’s inner tiles have shorter main colliders, which makes them unwalkable for the outer tiles:

A picture containing screenshot, font, line, design

Description automatically generated

Pic 16 – Tunnel. Inner tiles “unwalkable” for the outer tiles.

To move through the tunnel, the outer tile of the tunnel should be descended for its trigger to reach the first tunnel’s tile collider so it could be returned as a next walkable tile. After player crouches inside the tunnel, player will follow the tunnel’s design according to the grid:

A picture containing screenshot, font, line, text

Description automatically generated

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

* Moving platforms

Moving platforms simply utilize the Tile grid unit. Player can only step on and off the platform if standing on a grid tile which is in a close proximity to the platform to be able to identify it as an adjacent tile:

A screenshot of a computer

Description automatically generated with low confidence

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

A screenshot of a computer

Description automatically generated with low confidence

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

Vertical moving platform could be designed in a similar way. Refer to the ample scene for the example of the implementation of the vertical moving platform.

# Script reference.

Navigation

Classes

TileCore

Description

Defines a tile. Each Tile game object should have this component. It has obstacle collider which will be triggered when an “Obstacle” tag has been detected. Tile is “unwalkable” when an obstacle is detected.

Properties

|  |  |
| --- | --- |
| obstacleTrigger | refers to the TriggerOnTag component which will have “IsTriggered” property 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 tileCollider and colliderDefaultSize. |
| Update | adjusts tileCollider’s size depending onobstacleTrigger’s “IsTriggered” property. |

TileController

Description

Provides functionality that is responsible for a grid-based movement. Has 4 triggers that represent 4 directions. Returns an adjacent tile object when the player is moving in a specified direction.

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 adjacent tile depending on the provided parameter “direction”. |

Private Methods

|  |  |
| --- | --- |
| ForwardTile | returns TileController component of the adjacent tile in the forward direction if the forwardTrigger’s property “IsTriggered” is true, otherwise returns itself. |
| BackTile | returns TileController component of the adjacent tile in the back direction if the backTrigger’s property “IsTriggered” is true, otherwise returns itself. |
| LeftTile | returns TileController component of the adjacent tile in the left direction if the leftTrigger’s property “IsTriggered” is true, otherwise returns itself. |
| RightTile | returns TileController component of the adjacent tile in the right direction if the rightTrigger’s property “IsTriggered” is true, otherwise returns itself. |

Enumerations

Direction

Description

Keeps directions with values of their Euler degrees.

Properties

|  |  |
| --- | --- |
| Forward | Keeps value of 0 degrees |
| Left | Keeps value of 270 degrees |
| Back | Keeps value of 180 degrees |
| Right | Keeps value of 90 degrees |
| Round | Keeps value of 360 degrees |

Player

Classes

PlayerCore

Description

Defines movement keys. Requires PlayerController component.

Properties

|  |  |
| --- | --- |
| moveForward | defines the key code for the step forward |
| moveBack | defines the key code for the step back |
| moveLeft | defines the key code for the step left |
| moveRight | defines the key code for the step right |
| turnLeft | defines the key code for the turn left |
| turnRight | defines the key code for the turn right |

Private Methods

|  |  |
| --- | --- |
| Start | initializes the PlayerController component. |
| Update | executes PlayerController component’s methods depending on the key pressed. |

PlayerController

Description

Uses the grid-based tile map to perform movement and rotation functionality.

Properties

|  |  |
| --- | --- |
| smoothTransition | defines wather player’s movement and rotation instant or not. |
| movementSpeed | defines movement speed. |
| rotationSpeed | defines rotation speed. |
| precision | defines how precise player should be placed on the Tile’s position. |
| grid | defines the start tile of the grid. |
| IsBusy | is true when player is moving or rotating. |

Fields

|  |  |
| --- | --- |
| currentDirection | direction player’s camera is facing. Defined by Enumeration “Direction” |
| targetTile | Next tile player should move towards. |
| TargetRotation | rotation player should rotate towards. |

Private Methods

|  |  |
| --- | --- |
| Start | initializes currentDirection field as “Forward”. Initializes targetTile field as grid parameter. Initializes targetRotation based on currentDirection value. |
| Update | executes Move and Rotate methods if IsBusy property is true (targetTile assigned but not yet reached or targetRotation changed but not yet reached). |
| Move | moves player towards the targetTile’s position. |
| Rotate | rotates player towards targetRotation’s Euler angles. |
| LocalDirection | returns the direction relative to the current direction based on provided parameter. (ex: currentDirection is Left, parameter direction is Left, returns Back) |

Public Methods

|  |  |
| --- | --- |
| MoveTowards | reassigns a new targetTile based on provided “direction” parameter relative to the current direction. |
| RotateTowards | reassigns currentDirection based on provided “direction” parameter relative to the current direction. |

Helpers

Classes

TriggerOnTag

Description

A helper component that has a state “IsTriggered” when detecting a specified tag. All Tags should be specified in a Tag enum. Used by TileController to identify adjacent tiles, a tag “Tile” is specified for that. Can 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. True if gameobject’s collider intersects a collider of the object with the specified Tag. |
| TriggerObject | a game object with the specified tag that intersects this component’s game object. |

Private Methods

|  |  |
| --- | --- |
| OnTriggerEnter | sets “IsTrigger” to True if interacts with the object of specified Tag. And saves the object as the TriggerObject property. |
| OnTriggerExit | sets “IsTrigger” to False if stops interacting with the object of the specified Tag. And assigns null to the TriggerObject. |

Enumerations

Tag

Description

Keeps tags that could be referenced in code.

Level

Classes

HiddenPathController

Description

Hides/reveals the path by elevating the main collider over the obstacle collider of the walkable tile.

Properties

|  |  |
| --- | --- |
| maxElevation | y value of relative elevation for the main collider. |
| revealPath | the Key Code that triggers Hiding/Revealing the path. |
| revealTileModel | A model of the tile to be revealed. |

Private Methods

|  |  |
| --- | --- |
| Start | initializes a trigger on tag component, reveal collider (main collider assigned to the object that is tagged as “Obstacle”) and the default reveal collider position. |
| Update | holds functionality for hiding and revealing the path. |

CrouchController

Description

CrouchController component descents the Tile’s position which this component assigned to.

Properties

|  |  |
| --- | --- |
| crouch | the Key Code that triggers the Descent functionality. |
| descension | y value of relative descension of the Tile’s position. |

Private Methods

|  |  |
| --- | --- |
| Descent | Descents the game object’s position for the given descension value provided in the component’s properties. |