Aggregation Diagram

Graphical user interface, application, Teams

Description automatically generated

Application

Description automatically generated with medium confidenceDiagram

Description automatically generated with medium confidenceDiagram

Description automatically generated with medium confidenceA picture containing application

Description automatically generatedInheritance Diagram

Text

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generatedClass Diagram

Text

Description automatically generated

Shape

Description automatically generated with low confidence

Text

Description automatically generated with medium confidenceA picture containing text

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generatedA picture containing text

Description automatically generatedText

Description automatically generated with medium confidenceGraphical user interface, text, application

Description automatically generatedText

Description automatically generated with low confidenceA picture containing shape

Description automatically generatedA black screen with white text

Description automatically generated with low confidence%3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20value%3D%22PlayerGroundedState%22%20style%3D%22swimlane%3BfontStyle%3D2%3Balign%3Dcenter%3BverticalAlign%3Dtop%3BchildLayout%3DstackLayout%3Bhorizontal%3D1%3BstartSize%3D26%3BhorizontalStack%3D0%3BresizeParent%3D1%3BresizeLast%3D0%3Bcollapsible%3D1%3BmarginBottom%3D0%3Brounded%3D0%3Bshadow%3D0%3BstrokeWidth%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22800%22%20y%3D%22600%22%20width%3D%22140%22%20height%3D%2260%22%20as%3D%22geometry%22%3E%3CmxRectangle%20x%3D%22230%22%20y%3D%22140%22%20width%3D%22160%22%20height%3D%2226%22%20as%3D%22alternateBounds%22%2F%3E%3C%2FmxGeometry%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%223%22%20value%3D%22move\_input%3A%20Vector2D%22%20style%3D%22text%3Balign%3Dleft%3BverticalAlign%3Dtop%3BspacingLeft%3D4%3BspacingRight%3D4%3Boverflow%3Dhidden%3Brotatable%3D0%3Bpoints%3D%5B%5B0%2C0.5%5D%2C%5B1%2C0.5%5D%5D%3BportConstraint%3Deastwest%3B%22%20vertex%3D%221%22%20parent%3D%222%22%3E%3CmxGeometry%20y%3D%2226%22%20width%3D%22140%22%20height%3D%2226%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%224%22%20value%3D%22%22%20style%3D%22line%3Bhtml%3D1%3BstrokeWidth%3D1%3Balign%3Dleft%3BverticalAlign%3Dmiddle%3BspacingTop%3D-1%3BspacingLeft%3D3%3BspacingRight%3D3%3Brotatable%3D0%3BlabelPosition%3Dright%3Bpoints%3D%5B%5D%3BportConstraint%3Deastwest%3B%22%20vertex%3D%221%22%20parent%3D%222%22%3E%3CmxGeometry%20y%3D%2252%22%20width%3D%22140%22%20height%3D%228%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3C%2Froot%3E%3C%2FmxGraphModel%3E

A picture containing text

Description automatically generated

Text

Description automatically generated with low confidence

A picture containing text, screenshot

Description automatically generated

Text

Description automatically generated with medium confidence

Text

Description automatically generated

A picture containing shape

Description automatically generated

Text

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated with low confidenceA picture containing graphical user interface

Description automatically generated

Text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generatedText

Description automatically generatedText

Description automatically generated

A picture containing shape

Description automatically generated

Text

Description automatically generated with medium confidence

A black rectangle with white text

Description automatically generated with low confidence

Text

Description automatically generatedText

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated with low confidenceText

Description automatically generated with medium confidence

Shape

Description automatically generated with low confidence

A picture containing shape

Description automatically generatedText

Description automatically generated with medium confidence

Text

Description automatically generated with low confidenceA picture containing shape

Description automatically generatedA picture containing shape

Description automatically generated

Algorithms

Linear Interpolation: This algorithm will interpolate between two values using an interpolant. This is used to find a point some fraction of an amount along a line between two endpoints.

*function* lerp(flt\_1, flt\_2, lerp\_amount) *do*

*return* ((flt\_1 \* (1 - lerp\_amount)) + (flt\_2 \* lerp\_amount))

*end*

Sign: This algorithm returns the sign of an input. If the input is greater than or equal to 0 then 1 is returned. Otherwise, zero is returned.

*function* sign(flt) *do*

*if* flt >= 0 *do*

*return* 1

*end*

*else* *do* *return*

*return* -1

*end*

*end*

RigidBody2D.AddForce: This algorithm uses the physics equation f=ma to apply a force to a physics body.

I had first seen this algorithm used in Unity and I have researched how ForceMode is used and I have implemented this algorithm partially in Python as I have only used the ForceModes Force, Impulse and Acceleration in my code.

The function has arguments:

* Force: This is represented using a 2D vector
* DeltaTime: The time difference between when the previous frame that was drawn and the current frame was drawn
* ForceMode: Dictates the way that the force is applied to the body

Graphical user interface

Description automatically generated

Camera.UpdateScroll: This function updates the x scroll value and the y scroll value for the camera. This ensures that the player is always on the screen when playing.

*set* scroll\_x = scroll\_x + int((entity\_x - scroll\_x  - int((display\_surface\_width + target\_rect\_width) / 2)) \* 0.2)

*set* scroll\_y = scroll\_y + int((entity\_y - scroll\_y  - int((display\_surface\_height + target\_rect\_height) / 2)) \* 0.2)

*set* scroll\_x =  min(max(0, scroll\_x), level\_width - display\_surface\_width)

*set* scroll\_y =  min(max(0, scroll\_y), level\_height - display\_surface\_height)

Level.GenerateLevelData: This function generates a LevelData object which is a typed dictionary. This contains a list of LevelTile objects, the position of the player and a LevelFinish object.

The LevelData object is generate from a level section in an LDTK JSON file.

*set* tiles = []

*set* player\_position = [-1, -1]

*foreach* (layer in layer\_instances) *do*

*switch* (layer\_identifier) *do*

*case* "AutoTiles":

*foreach* (tile in auto\_tile\_layer) *do*

*//generate\_level\_tile*

                tiles.append(level\_tile)

*end*

*break*

*case* "Tiles":

*foreach* (tile in tile\_layer) *do*

*//generate\_level\_tile*

            tiles.append(level\_tile)

*end*

*break*

*case* "Entities":

*foreach* (entity in entity\_instances) *do*

*switch* (entity\_instance\_identifier) *do*

*case* "Player":

*set* player\_position = entity\_position

*break*

*case* "Finish":

*//generate\_level\_finish\_object*

*set* level\_finish = level\_finish

*break*

*end*

*end*

*end*

*end*

One part of the `player\_run\_state` update function calculates the force necessary to apply to the player.

This works by first calculating the speed the player is trying to reach (target\_speed) and then calculating an acceleration rate based on the target speed.

The speed difference is then calculated and the acceleration rate is multiplied by the speed difference.

This means that if the difference between the player’s current x velocity is close to the target speed then only a small force is applied.

An extra force is also applied to decrease the time it takes to change direction.

*set* target\_speed = x\_move\_input \* player\_run\_speed

*set* speed\_difference = target\_speed - player\_x\_speed

*if* abs(target\_speed) > 0 *do*

*set* acceleration\_rate = player\_acceleration\_rate

*end*

*else* *do*

*set* acceleration\_rate = player\_deceleration\_rate

*end*

*set* movement = pow(abs(speed\_difference) \* acceleration\_rate, self.player.velocity\_power) \* sign(speed\_difference)

*//add the force the player with ForceMode FORCE and pass in delta\_time*

*if* abs(target\_speed) == 0 *do*

    friction = sign(player\_x\_speed) \* min(abs(player\_x\_speed), self.player.friction

    )

*//add the friction force to the player with ForceMode IMPULSE*

*end*