Aggregation Diagram

Graphical user interface

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 shape

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

Text

Description automatically generated with medium confidence

A black screen with white text

Description automatically generated with low confidenceGraphical user interface

Description automatically generated with low confidence

Shape

Description automatically generated with low confidence

A picture containing shape

Description automatically generatedText

Description automatically generated with medium confidence

A picture containing shape

Description automatically generatedText

Description automatically generated with low confidenceA 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(a, b, lerp\_amount) *do*

*return* ((a \* (1 - lerp\_amount)) + (b \* 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.

The tiles are placed into a dictionary where the key is tuple that signifies the chunk co-ordinate that the tile is in.

The position of the player and a LevelFinish object.

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

The chunk co-ordinate of a tile is determined by the following algorithm:

chunk\_x = tile\_rect\_x // chunk\_size

chunk\_y = tile\_rect\_y // chunk\_size

*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*

Data Dictionary

|  |  |  |
| --- | --- | --- |
| Variable Name | Data Type | Use |
| config | NeaGameConfig | Stores the configuration for the game |
| nea\_game | NeaGame | Creates the root window for the application that controls which window is shown each frame |
| screen | pygame.Surface | The screen that the display\_surface is scaled onto. |
| display\_surface | pygame.Surface | A smaller surface where tiles and entities are rendered. This surface is then scaled up to the size of the screen |
| engine | Engine | Provides an interface to update and access the time difference between when the previous frame was drawn and when the current frame was drawn |
| clock | pygame.time.Clock | Allows functionality to control the framerate of the application |
| previous\_time | float | Stores the timestamp of when the previous frame was drawn |
| delta\_time | float | The time difference between when the previous frame was drawn and when the current frame was drawn |
| fps | int | Stores the fps that application is trying to run at |
| sound\_manager | SoundManager | Creates functionality for controlling the volume and sounds the background music and sound effects. |
| sound\_effects | dict[str, pygame.mixer.Sound] | Stores a dictionary of the each sound effect |
| bgm\_volume | float | Stores the volume of the background music as a float between 0 and 1 |
| sfx\_volume | float | Stores the volume of all sound effects as a float between 0 and 1 |
| current\_bgm | pathlib.Path | Stores the path to the current background music |
| sound | pygame.mixer.Sound | Stores the sound to be played on the current frame |
| active\_window | Window | Stores the active window |
| scale\_factor | int | Stores the ratio between the resolution of the screen and the resolution of the display\_surface |
| windows | dict[str, Window] | Stores each window that has been active. |
| transition | None | Callable[[], None] | Stores the current transition where the transition is a function that has no arguments and returns no value |
| transition\_start\_radius | int | Stores the starting radius of the transition |
| transition\_circle\_centre | tuple[int. int] | Stores the centre of the circle for the transition |
| current\_transition\_frame | int | Stores the frame that the current transition is on |
| is\_transitioning | bool | Stores whether the application is transitioning or not |
| is\_transition\_done | bool | Stores whether the current transition is done |
| events | list[pygame.event.Event] | Stores a list of each event that has occurred on the current frame |
| surface | pygame.Surface | The surface that the transition is drawn onto, a colourkey is then used to remove the rest of colour of the surface |
| parent | NeaGame | Stores the main application for the window |
| buttons | dict[str, Button] | Stores a dictionary of the buttons for the window |
| passive\_image | pygame.Surface | Stores the image of the button when it is not hovered over |
| active\_image | pygame.Surface | Stores the image of the button when hovered over but not clicked |
| on\_click\_image | pygame.Surface | Stores the image of the button when it is clicked |
| current\_image | pygame.Surface | Stores the current image of the button |
| rect | pygame.Rect | A Rect object for the object |
| mask | pygame.Mask | A Mask object for the object, which can be used for pixel perfect collision detection |
| can\_be\_clicked | bool | Whether the button can be clicked or not |
| clicked | bool | Whether the buttons is clicked or not on the current frame |
| click\_delay | float | The time delay between when the button is clicked and the clicked variable changes. This is used to be able to show the click animation for the button |
| click\_timer | float | The time since the button was clicked |
| mouse\_pos\_in\_mask | tuple[int. int] | The position of the mouse in the mask |
| background\_layers | list[BackgroundLayer] | Stores a list of each layer that makes up the background |
| image | pygame.Surface | Stores the image of the object |
| sine\_scale\_factor | float | The value which the sine curve for the background layer is scaled by |
| sine\_stretch\_factor | float | The value which the sine curve for the background layer is stretched by |
| sine\_translation\_factor | float | The value which the sine curve for the background layer is translated by |
| x\_scroll | float | The value which the background is scrolled by on the x-axis |
| handle\_image | pygame.Surface | A copy of the image for the layer |
| clip\_rect | pygame.Rect | The subsection of the layer which will be used for the background |
| splash\_screen\_opacity | float | The opacity of the background between 0 and 1 |
| title | Title | The title for the main menu |
| scaled\_mouse\_pos | tuple[int, int] | Stores the scaled mouse position of the mouse in the display surface |
| mouse\_clicked | bool | Stores whether the mouse has been clicked on the current frame |
| window | Window | Stores the window to be switched to |
| button\_folder\_path | pahtlib.Path | Stores the path to the button folder for the window |
| padding | tuple[int, int] | The minimum distance between the button and the edge of the screen |
| spacing | tuple[int, int] | The distance between each button on both axis |
| world\_num | int | The world number of the level to play |
| level\_num | int | The level number of the level to play |
| world\_identifier | str | The identifier of the world |
| level\_identifier | str | The identifier of the level |
| world | World | Stores the world as a world object |
| tileset | Tileset | Stores the tileset for the world |
| grid\_height | int | The height of tileset image in tiles |
| grid\_width | int | The width of the tileset image in tiles |
| grid\_size | int | The size of the tiles in pixels |
| collision\_types |  |  |