Windows User

This document contains all classes and their respective members, within my implementation of the Game for AE2. A report on testing of this game is also included within this report.

CGP AE2 Implementation and Testing

With appropriate code documentation

Contents

[Class Documentation 2](#_Toc471291652)

[Class Diagram 2](#_Toc471291653)

[Structures and Enumerations 3](#_Toc471291654)

[Class Member Details 4](#_Toc471291655)

[GameManager 4](#_Toc471291656)

[InputManager 5](#_Toc471291657)

[GameBitmap 5](#_Toc471291658)

[GameEntity 6](#_Toc471291659)

[GameLevel 7](#_Toc471291660)

[Enemy 7](#_Toc471291661)

[Player 8](#_Toc471291662)

[GameCollisionSystem 8](#_Toc471291663)

[Level Layout 9](#_Toc471291664)

# Design Alterations

Instead of using bounding boxes to check for collision, values for where mobile-GameEntities are allowed to move (in both X and Y), are calculated after the level is generated. This results in increased performance, as these values are cached, meaning that the game would not have to check for collision whilst a GameEntity is moving (but the game will take longer to load).

Before implementation, the system was intended to calculate X and Y values for any level design, but only if the valid positions for each column and row, were continuous, without breaks in the ranges by invalid blocks and only one value range per column and row.

Into implementation, this is true for the X-values, but for the Y-values; the Player can warp from any row, to the row they start on, by attempting to move into a blocking entity on the row below they row they are currently upon.

Ultimately; if the Player is careful in their movement, they can reach the end of the level, with a subtle, but at least slight increase in frame rate, as movement values are cached beforehand. In reflection upon the implementation of this system though, the Programmer should have allotted more time to the implementation of this sub-system, as an alternative to bounding-box collision, as they had not anticipated the issues that would occur, with the implementation of such a system.

# Class Documentation

## Class Diagram

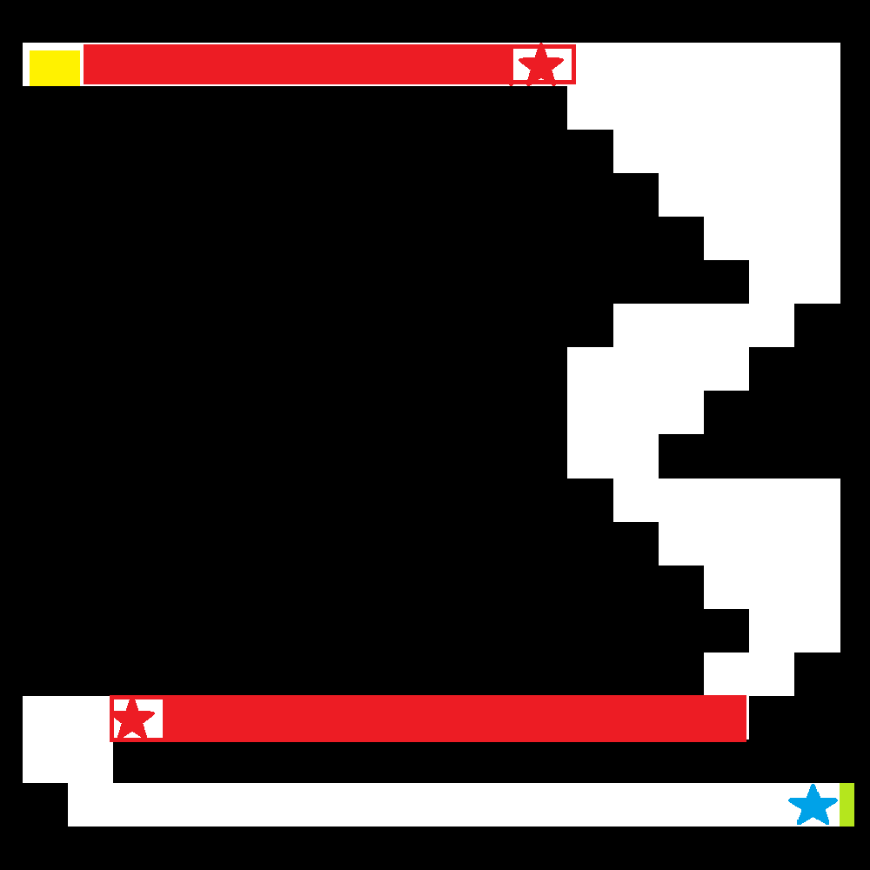
For all the classes in this game:



## Level Layout

The level layout, for basic implementation of the intended features, is as follows (first version):

Where black indicates wall tiles, yellow indicates the key for level-exit-door, green indicates the level-exit-door, blue indicates the Player’s starting point on this level and Red indicates initial Enemy positions, as well as their patrol routes (on the next version of this diagram).



The next iteration of the level; is shown to the left:

This has far greater equivalency to the level in implementation, given the alterations I put in place, to account for mechanical restrictions (namely; how high the Player can jump, as well as the degree of air-control they will have).

The level-exit door, the key for that door, as well as the Player and the point at which they spawn into the level, will received implementation in the next iteration (but are still shown as reference to the intended placement locations in the level).

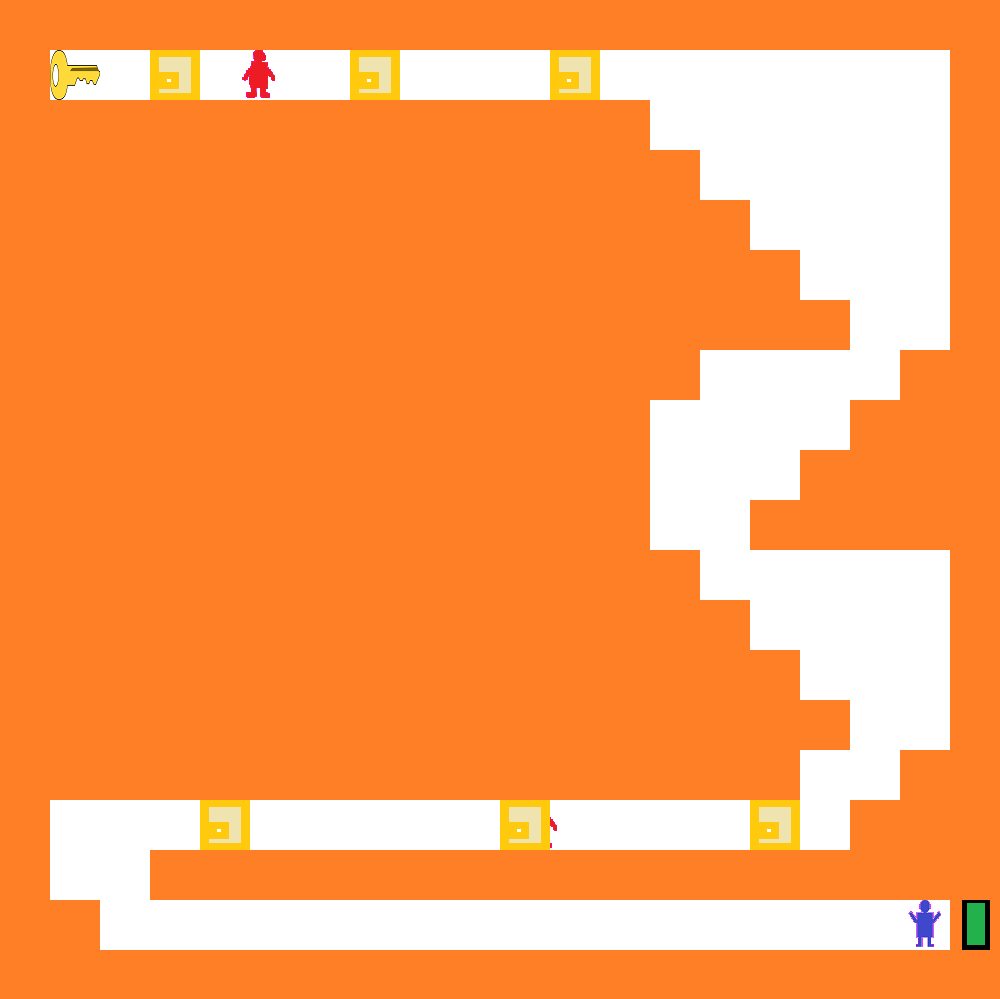
# Testing

For all the features intended for implementation in the game, testing for such, is as detailed in the table below:

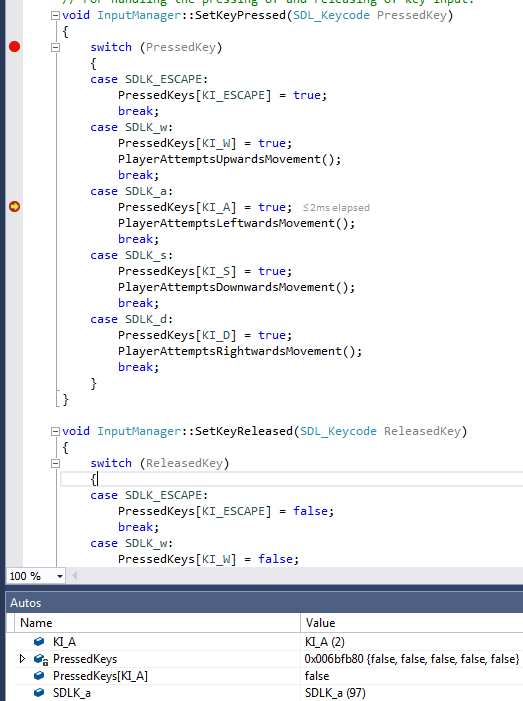
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Expected Results | Actual Results | Successful? |
| 1 | Preconditions: The game is running, there is an empty space (or Enemy Door) to the left of the Player. The Player presses the A key. | The Player’s bitmap is moved by MOVEMENT\_SPEED pixels, leftwards. | The Player’s bitmap is moved by MOVEMENT\_SPEED pixels, leftwards. | True. |
| 2 | Preconditions: The game is running, there is an empty space (or Enemy Door) to the right of the Player. The Player presses the D key. | The Player’s bitmap is moved by MOVEMENT\_SPEED pixels, rightwards. | The Player’s bitmap is moved by MOVEMENT\_SPEED pixels, rightwards. | True. |
| 3 | Preconditions: The game is running, there is an empty space (or Enemy Door) above the Player. | The Player’s bitmap is moved by MOVEMENT\_SPEED pixels, upwards. | The Player’s bitmap is moved by MOVEMENT\_SPEED pixels, upwards. | True. |
| 4 | Preconditions: The game is running, there is an empty space (or Enemy Door) below the Player. | The Player’s bitmap is moved by MOVEMENT\_SPEED pixels, downwards. | The Player’s bitmap is moved by MOVEMENT\_SPEED pixels, downwards. | True. |
| 5 | Preconditions: The game is running, there is NOT an empty space (or Enemy Door) to the left of the Player. The Player presses the A key. | The Player’s Bitmap remains at its current position. | The Player’s Bitmap remains at its current position. | True. |
| 6 | Preconditions: The game is running, there is NOT an empty space (or Enemy Door) to the right of the Player. The Player presses the D key. | The Player’s Bitmap remains at its current position. | The Player’s Bitmap remains at its current position. | True. |
| 7 | Preconditions: The game is running, there is NOT an empty space (or Enemy Door) above the Player. | The Player’s Bitmap remains at its current position. | The Player’s Bitmap remains at its current position. | True. |
| 8 | Preconditions: The game is running, there is NOT an empty space (or Enemy Door) below the Player. | The Player’s Bitmap remains at its current position. | The Player’s Bitmap remains at its current position, UNLESS the Player is on a row above the row they start the level on, at which point, they will warp to the row they started upon | False. |
| 9 | Preconditions: The Player is on the same row as an Enemy, the Player has collided with this Enemy. | The Player loses a life and respawns at the level starting point. | The Player clips through this Enemy. | False. |
| 10 | Preconditions: An Enemy is on the same row as the Player, the Enemy has collided with the Player. | The Player loses a life and respawns at the level starting point. | This Enemy clips through the Player. | False. |
| 11 | Preconditions: The Player has not collected the Key. The Player collides with the Key. | The Level Exit Door opens. | The Level Exit Door remains in its current state. | False. |
|  |  |  |  | False. |
|  |  |  |  | False. |

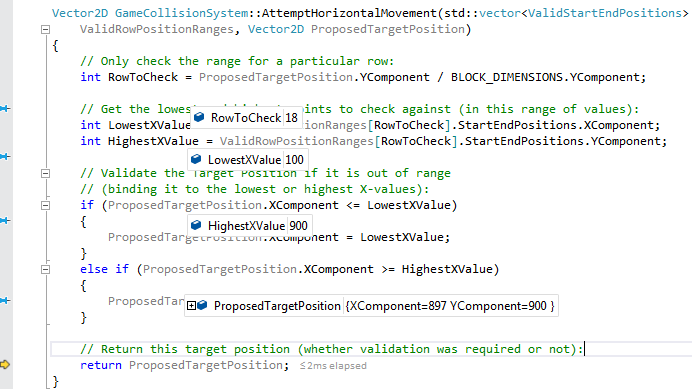
## Evidence

### Tests 1-8

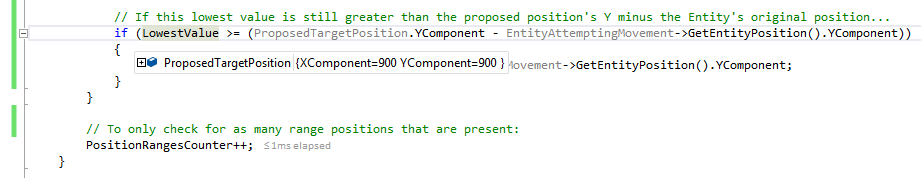
When the game begins, the level is shown as per its design (with the Player in blue, Enemies in red, the key in the top left, and the level-exit-door in green (at the bottom right of the level)):

If the Player attempts to move in any direction (see next page):

Methods are then called, which cause the Player to move upwards, leftwards, rightwards or downwards, if they can (see below):

So in this case, the Player is able to move to the left (XComponent of the proposed position, is equal to their current position minus MOVEMENT\_SPEED, which is less than the highest allowed x value).

This will also result in successful (horizontal movement), rightwards, if possible, as well as a similar method for vertical movement (next page) resulting in successful downwards or upwards movement:



### All other tests

Due to time constraints, evidence to verify the results of these tests, is not referenced here (Testing should have received execution earlier in development, as well as its documentation).