Valkan

Lourens Verhage
6762115

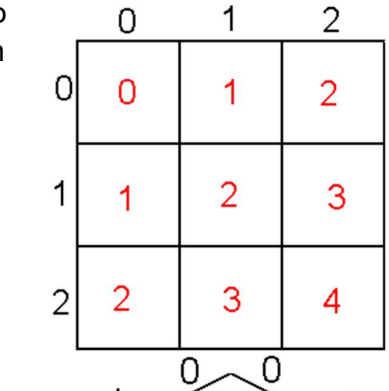
2.5D Template

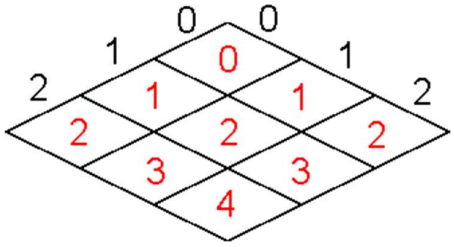
2.5D is more difficult to make than a flat 2D game. 2.5D is a 2D game with the aesthetics of a 3D game. The most difficult part of a 2.5D game is the drawing order of the objects. The grid will have to be rotated and flattened in order to create the desired aesthetics. To create the desired 3D effect the tiles have to be drawn in the right order.

Technical design

1. 2.5D grid

To make a 2.5D grid out of a 2D grid a matrix will have to be used. In order to get back to the grid coordinates from the game coordinates the inverse of this matrix will be used. The tiles will have a size ratio of 2x1. This size ratio is chosen because it makes for much easier calculations.



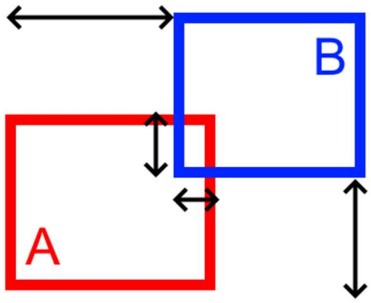


2. Draw

To make sure the tiles and entities on top of them are drawn in the right order every tile on the grad has a list with passengers. Passengers are entities that are un top of the tile. The tiles are drawn in the order of the red numbers in the images. This makes sure that walls in front of the player will be draw over the player. When a tile is draw the entities on top of the tile are drawn directly afterwards.

To ensure that the game does not draw anything that is not on the screen only tiles inside the screen are drawn and therefore also only entities inside the screen are drawn. To make sure that the program does not have to check every tile if it is on screen or not the draw will first calculate the grid coordinate of the tile that is just outside the screen. It will than only draw/check the tiles from that coordinate to the tiles just outside the other end of the screen. This ensures that a lot less objects have to be checked and helps the performance a lot.

3. Collision detection



Only entities that move will perform collision detection. Because when a collision happens at least one object has to have moved this will be enough to ensure that every collision will be detected. Collision detection will not be done around the entire field, it will only be done on the nearby tiles around the entity. Collision detection is done through rectangles. When two rectangles intersect the intersection depth will be calculated and one of the objects will be moved back the shortest intersection depth. Apart from moving the objects back a specific distance the program could also respond in different ways like when a sword hits a monster the monster will lose health, or when a player walks against a barrel the barrel will move.

4. Location update

Only updating entities that are in a specific range around the screen ensures that no entities at the other end of the map will be calculated when it is not needed. This is done in a similar way as drawing the tiles on screen.