|  |
| --- |
| [Type the company name] |
| Dungeon game |
| Homemade dungeon game |

|  |
| --- |
| Hans  8/4/2012 |

# Background

Since there is no Dungeon Keeper 3 it was decided to try to create a similar game, although simple but might still as a good exercise on game programming.

The game will be tile-based, starting as a 2d project to move to a 3d engine at a later stage, where user input will manipulate, mining the tiles, the map to a certain degree. All interaction between player units and fully AI controlled unit will be automatic.

## Development

The development will be done using Visual Studio 2010 Ultimate in conjunction with XNA Game Studio 4.0. Handling of files will be done using a local GiT repository.

# Requirements

## Functional Requirements

Some functional requirements are needed to be able to have all functionality needed

1. Customizable map sizes: Be able to change the size of the map without implications.
   1. Only redraw tiles that have been changed.
   2. Possible to generate new maps.
      1. Generated maps should be random with some consistency to avoid cluttering it in too much.
   3. Maps should be read from an external file
2. Perspective camera: Implement a perspective camera to enable size of map to extend outside the window.
   1. Separate files to enable change to 3d engine.
3. 2D graphics: Simple graphical tiles for testing purpose.
4. 3D graphics: 3D models to use as tiles/blocks and creatures.
5. AI.
   1. Spawn point for AI players.
   2. Player controlled AI creatures.
      1. If adjacent tile is marked and minable, mine it if possible.
   3. AI battles (If adjacent to each other). See FR8.
   4. Multiple spawn point for AI players.
6. Creature XP bars: Each creature can earn experience from digging/fighting to get higher stats.
   1. Stats, name and leveling increments should be read from an external file
7. Different creatures
   1. Melee with high defense but low resistance and medium damage
   2. Range with low resistance and defense and medium/high damage
   3. Melee with medium defense and resistance
   4. Melee with high damage but low resistance and defense
8. Tiles
   1. Several intractable tiles will be available
      1. Upon clicking on a “minable” tile it will be highlighted.
      2. Clicking on a highlighted tile will unhighlight it.
   2. Mining a gold tile will generate a gold leftover
      1. Several values will be available
9. Battle
   1. If enemy is in adjacent tile initiate battle.
      1. Battle order is determined by Haste attribute.
         1. Highest first.
         2. If more than twice the value hit twice.
      2. Determine the outcome based on attributes.
         * 1. Max value to avoid healing in case of extreme defense.
           2. Same equation done for both sides in the battle, only need one battle function in that case.
   2. Attack initiates each second based on gameTime. See Battle SSD.
   3. Each creature has none or some resistance.
      1. Increase the complexity by having different kind of attacks and resistance
   4. Each creature has a range, if that range is ≥ 1 it can shoot over several squares.
      1. Add to be able to attack diagonally
10. Resources
    1. Mining gold gives resources needed to buy new creatures
       1. Add other resources as well such as diamonds
          1. Same resource but more valuable
    2. Cap on amount of creatures in total
       1. The better creature the more resource cap it requires
11. Research to be able to get new creatures and such.
    1. User selects a skill/creature to research and waits for its completion
       1. Costs resources
       2. Give creatures better stats
       3. Creatures cost less
       4. Research to generate small amount of resources
    2. Base on same kind of XML file as creatures
12. Sentry guns/traps
    1. Research traps and sentry guns to place out on the map to guard chokepoints
       1. Similar to Orcs must Die™

## Design choices

1. Handling Creatures and Enemies
   1. Two different ways can be used to handle the setup on data structures for the creatures (player controlled) and enemies (AI controlled). Either each tile contains a creature and/or enemy, which will be checked during the redraw phase when all tiles, will be checked anyway for status change. The benefit of this is that all creatures/enemies will be tested and it’s easy to find adjacent enemies/creatures due to the grid network of the tiles. The second option is to have a list with pairs of position and creature/enemy data. This will result in more comparisons to find the closet/adjacent adversary. Every combat turn this will result in N\*M comparisons where N and M are length of the creature respectively enemy lists. The first alternative will result in only 4\*N comparisons (not possible to attack diagonally) where N is the number of creatures. This should be the faster alternative but results in limitation in the program, no ranged creatures/enemies possible.
   2. When using the two lists instead of checking each tile it will be possible to check weather a creature/enemy can range-hit and adversary by taking a vector from A to B and checking if it intersects with another object. Only possible in the 3D case, might be possible in 2D case as well if all impassable tiles have a specific rectangle and check the ray if it intersects with one of those; a lot more work and will be unfeasible at the moment due to the short time the 2D case will persists.
   3. Pseudo-code for list traversing
      1. Foreach creature in creatures  
          Foreach enemy in enemies  
          If adjacent(creature.pos, enemy.pos)  
          Creature.battle(enemy)  
          If enemy.hp <= 0  
          Enemies.delete(enemy)

This battle will be resolved in reversed order as well giving the enemy chance to attack as well. If the enemy/creature reach zero in hp they will be removed from the list.

## System Sequence Diagram

System sequence diagrams of the standard flow of moving and battling the creatures and enemies both on a high level as well as a detailed state.

### Standard turn

SSD of a standard turn, move battle and so on, high level

### Move turn

How the creatures move

### Battle turn

More in detail about the battle