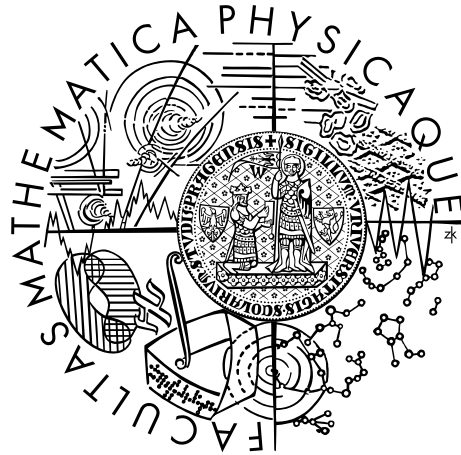


Charles University in Prague
Faculty of Mathematics and Physics

BACHELOR THESIS



Jaroslav Jindrák

The Dungeon Throne: A 3D Dungeon Management Game

Department of Distributed and Dependable Systems

Supervisor of the bachelor thesis: Mgr. Pavel Ježek, Ph.D

Study programme: Computer Science

Study branch: TODO: IPSS?

Prague 2016

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In date

signature of the author

Title: The Dungeon Throne: A 3D Dungeon Management Game

Author: Jaroslav Jindrák

Department: Department of Distributed and Dependable Systems

Supervisor: Mgr. Pavel Ježek, Ph.D, Department of Distributed and Dependable Systems

Abstract: Abstract. TODO:

Keywords: RTS game dungeon management Lua scripting Ogre CEGUI

Dedication. TODO:

Contents

Introduction	4
1 The Game	5
1.1 Dungeon Managment Genre	5
1.2 Modifiability in Games	5
1.3 Similar Games	5
1.3.1 Dungeon Keeper	5
1.3.2 Garry's Mod	5
1.3.3 Minecraft	5
1.3.4 Dungeon Siege	5
1.4 Thesis Goals	5
2 Problem Analysis	6
2.1 Engine Design	7
2.1.1 Inheritance based (wording?)	7
2.1.2 Component based (wording?)	7
2.1.3 Entity Component System	7
2.2 Programming Language	7
2.2.1 C++	7
2.2.2 Lua	7
2.2.3 Java	7
2.2.4 C# ???	7
2.3 Scripting Language	7
2.3.1 Custom	7
2.3.2 Lua	7
2.4 Libraries	7
2.4.1 3D Rendering	7
2.4.2 Graphical User Interface	7
2.5 Algorithms	7
2.5.1 Pathfinding	7
2.5.2 Level Generation	7
2.6 Serialization	7
2.6.1 Binary	7
2.6.2 XML	7
2.6.3 Lua	7
3 Implementation / Dev Docs?	8
3.1 Engine (?Layout)	8
3.2 Game	8
3.2.1 State Design Pattern	8
3.3 Components	8
3.4 Systems	9
3.4.1 CombatSystem	9
3.4.2 EntitySystem	9
3.4.3 EventSystem	10

3.4.4	GridSystem	10
3.4.5	TaskSystem	10
3.4.6	WaveSystem	10
3.4.7	Other	11
3.5	Helpers	12
3.6	Script	12
3.7	LuaInterface	12
3.7.1	Initialising the API	13
3.7.2	Extending the API	13
3.8	GUI	13
3.8.1	Console	13
3.8.2	EntityCreator	13
3.8.3	EntityTracker	14
3.8.4	ResearchWindow	14
3.8.5	Other	14
3.9	SpellCaster	15
3.10	Utilities	15
3.10.1	Camera	15
3.10.2	Effects	15
3.10.3	EntityPlacer	15
3.10.4	LevelGenerator	16
3.10.5	RayCaster	16
3.10.6	SelectionBox	16
3.10.7	Conditions	16
3.11	Pathfinding	16
3.11.1	Grid	17
3.11.2	Algorithms	17
3.11.3	Heuristics	17
3.11.4	Path Types	17
3.12	Serialization	17
3.13	Player	18
3.14	Singleton Design Pattern	18
3.14.1	Script	18
3.14.2	GUI	18
3.14.3	Player	18
3.14.4	Grid	18
3.14.5	EntitySystem	18
3.15	Scripting	19
3.15.1	Initialisation	19
3.15.2	Entity Representation	19
3.15.3	Blueprints	19
3.15.4	Research	19
3.15.5	Spells	19
3.15.6	Creating a Mod (? Modification)	19
3.16	Extending the Game	20
4	User's Documentation	21
4.1	wubwubwub	21

Conclusion	22
Bibliography	23
List of Figures	24
List of Tables	25
List of Abbreviations	26
Attachments	27

Introduction

1. The Game

An example citation: Anděl [2007] TEST

1.1 Dungeon Managment Genre

1.2 Modifiability in Games

1.3 Similar Games

1.3.1 Dungeon Keeper

TODO: Design influence.

1.3.2 Garry's Mod

TODO: Modifiability influence. (Mainly game mechanics.)

1.3.3 Minecraft

TODO: Modifiability influence. (Mainly items.)

1.3.4 Dungeon Siege

TODO: ECS influence.

1.4 Thesis Goals

2. Problem Analysis

2.1 Engine Design

2.1.1 Inheritance based (wording?)

2.1.2 Component based (wording?)

2.1.3 Entity Component System

2.2 Programming Language

2.2.1 C++

2.2.2 Lua

2.2.3 Java

2.2.4 C# ???

2.3 Scripting Language

2.3.1 Custom

2.3.2 Lua

Bindings

2.4 Libraries

2.4.1 3D Rendering

OpenGL

DirectX

Ogre3D

2.4.2 Graphical User Interface

Ogre Overlay

CEGUI

2.5 Algorithms

2.5.1 Pathfinding

Dijkstra

Breadth-first Search

Depth-first Search

A*

Solution ??

2.5.2 Level Generation

3. Implementation / Dev Docs?

3.1 Engine (?Layout)

- describe Component - System - Helper - LuaInterface - ... relationship
- possibly create a picture
- describe the "workflow" of the game? or leave it to other sections?

3.2 Game

- !ask PJ: do I have to describe functions even if they have documenting comments?
- describe how the game is initialized
- explain OIS and Ogre callbacks (+ parent classes)
- brief description of its members vs just mentioning them vs ignoring them as they will be explained later?
- level creation (new_level && generate_empty_level)
- throne monitoring? (keeping throne ID so there is only one target of the enemies)
- possibly add a subsections describing Ogre and CEGUI initialisation

3.2.1 State Design Pattern

- describe the pattern
- describe the states of the game
- explain why the pattern is NOT used

3.3 Components

- go through components one by one with UML-like pictures describing their members and purpose
- explain why strings are moved?
- explain why components need default values for every parameter of their constructor?
- go through some more complex concepts of the components (align states, maybe forward reference blueprints, etc.)

3.4 Systems

- define systems
- mention common base class and the reason for inheritance (list of systems...)
- describe the component-system relationship (none, system-component, system-components, systems-component, ...)

3.4.1 CombatSystem

- describe the update logic
- describe entity querying
- describe effect application
- talk about the fabulousness of the query & effect design!
- talk about the run away mechanism?
- talk about runtime projectile creation?
- talk about line of sight and LoS wrt BB?

3.4.2 EntitySystem

- describe the update logic
- describe cleanup (and its delay and why it's necessary)
- id system (back then vs now)
- entity creation process
- component containers
- describe component manipulation (add, load, set, delete, delete now, ...)
- mention entity registration for the entity creator window
- describe function array (reason, jmp table bonus, etc.)
- explain the macros!
- describe why constructors are delayed (so that the lua function where it's create continues before constructor call)
- maybe talk more about component loading from lua? or leave that to the scripting part? or never mention it?

3.4.3 EventSystem

- describe the update logic
- describe event persistence/destruction
- describe how to make event one (successful) pass only
- targeted/area events and area event expansion
- delete component vs delete entity discussion
- describe handling (fixed C++ vs Lua)
- this should be first time we encounter update periods and time multipliers, explain!

3.4.4 GridSystem

- describe the update logic (+freed/unfreed and Grid relation)
- talk about grid graphics used for debugging?
- explain structure placement
- describe alignment checks

3.4.5 TaskSystem

- describe the update logic
- talk about busy state, processing tasks and checking if task is complete
- probably only forward reference the Lua part? or talk about it right here?
!ask PJ!

3.4.6 WaveSystem

- describe states and maybe mention why the State design patten wasn't used (too simple)
- describe the update logic of the different states (-i countdown, spawning, chilling)
- talk about the spawning mechanism, blueprint vector, spawn nodes
- talk about wave entity monitoring and wave ending
- talk about the wave table
- talk about the wstart and wend callbacks
- talk about endless mode
- possibly explain how to write a wave? or add that to the scripting part?

3.4.7 Other

- here explain in alphabetical order the remaining systems
- say why these are not as important (that is, not important to be described) as the others

AISystem

- describe the update logic
- mention update forcing (in relation to the update period mentioned earlier)

GraphicsSystem

- say how it's supposed to maintain all manual graphics logic but atm only does explosions (since it's the only manual graphics object)
- describe the update logic

HealthSystem

- describe the update logic
- mention configurable regeneration period

InputSystem

- describe the update logic
- explain initial purpose and why it's not used atm

ManaSpellSystem

- describe the update logic (both mana regen and entity spell casting)
- possibly talk about the relationship between player spells and entity spells

MovementSystem

- describe the update logic
- talk about move and checked_move (and can_move_to) and why checked is not used but can be from lua (and was intended for 1st person mode)

ProductionSystem

- describe the update logic
- explain how products are spawned and placed

TimeSystem

- describe the update logic (mention the multiple comps updated)
- explain time event handling (and use)
- explain time event advancements

TriggerSystem

- describe the update logic
- explain how triggering works and why it works that way (factions)
- explain the general trap concept?

3.5 Helpers

- describe what helpers are and how are they used
- forward reference the not-singleton status of EntitySystem?
- explain why they were created (use in Lua and optionally in C++ for better code readability)
- explain why they are slower than direct component manipulation
- explain component-helper relationship
- mention the general structure of a helper
- maybe talk about why they are namespaces and not classes? (+ namespace -i allows static class without any change basically)

3.6 Script

- talk about how it works (it's basically a wrapper facade)
- mention lpp::Exception
- explain the Lua C API (only informative? add a new section with a tutorial? tell the reader to read the Lua Programming Language 3rd edition book?)
- maybe go through some of the more complex functions?

3.7 LuaInterface

- talk about why it's needed (Lua needs static)
- talk about why it's centralised in one class?

3.7.1 Initialising the API

- explain the C++ function binding process
- explain the Lua module hierarchy
- explain other aspects besides function binding

3.7.2 Extending the API

- explain the general body of the interface functions (stack, return etc.)
- (super) small tutorial on how to add new interface functions

3.8 GUI

- talk about the GUI hierarchy
- explain initialization
- explain CEGUI button binding? maybe in general CEGUI manipulation?
- explain save/load file listing
- mention the GUIWindow class, base class of the following GUI windows
- explain why it has escape_pressed handler and does not use CEGUI handlers for that (would be needed for all subwindows, too much a hassle - also would probably disregard the conditional closing)

3.8.1 Console

- talk about how awesome it is during debugging
- explain how execution and printing works (+ the history concept?)
- explain why it does not execute lines but multi lines after the execute button is pressed

3.8.2 EntityCreator

- mention that currently it is used for entity placement and the creation part is supposed to be a graphical way to make entities that is not currently in the game
- explain how it works
- explain how it's used for testing
- maybe mention that this is where the registered_entity from EntitySystem is used

3.8.3 EntityTracker

- explain how tracking works
- talk about why it's useful
- mention the upgrade, exp convert and delete functionality?

3.8.4 ResearchWindow

- explain initialisation, how it works
- explain dummy_unlock and its relationship with serialization

3.8.5 Other

- say that these are generally not that complex and quite straight forward

BuilderWindow

- talk about building registration (how it relates to unlocking)
- talk about the assembly line (also used in spell casting window) and why it's good (lack of images -> buttons need to be big and readable)
- don't forget to mention that this is just a graphical front end to EntityPlacer (with price management etc.) as is EntityCreator (though that one ignores price, and has all unlocked even enemies)

GameLog

- just mention that it is basically the ingame chat posting info for the player
- also maybe again mention history (same as console)?

MessageToPlayerWindow

- talk about button renaming and action assignment
- then mention how to use this as a whole

OptionsWindow

- talk about actions, key bindings and video settings (+ how they are done?)

SpellCastingWindow

- talk about spell registration (how it relates to unlocking)
- just mention the assembly line (already explained in BuilderWindow)
- talk about the relationship with the SpellCaster and how spell casting works on this side (simple invoking of the spell caster and marking the spell as active if needed)

- don't forget to mention that this is just a graphical front end to SpellCaster basically

TopBar

- just say something about its purpose (monitoring of resources)

3.9 SpellCaster

- discuss the spell casting concept in the game
- mention the spell types
- explain why it's so damn cool (ease of spell creation, run time spell creation)
- forward reference the Lua spell structure, which is explained later in Scripting

3.10 Utilities

- just say that these are classes that are not directly part of the game world but are used as background tools that support the game

3.10.1 Camera

- talk about how it allows free/nonfree mode, resetting and backups
- maybe talk more about the free nonfree mode and how to toggle them (both keybind and command)

3.10.2 Effects

- talk about how awesome they are and allowed the creation of extensible effect application framework in CombatSystem
- explain how to write one
- mention that they are used as template arguments and as such should conform the given interface (also explain that)
- maybe illustrate their structure on one of them

3.10.3 EntityPlacer

- explain its purpose and how it works
- explain why only the graphics component data is used (so it's ignored by the game world and serializer, etc.)

3.10.4 LevelGenerator

- explain their purpose and how to write one
- mention the cycle count constructor parameter
- mention the default tables in config (and that it's easy to add new ones)

RandomLevelGenerator

- say that this is just naive algorithm for a pseudo random level generation using number of neighbours that are gold deposits to determine if a gold deposit should be spawned

3.10.5 RayCaster

- explain that I'm a noob that stole other programmer's idea for this from the Ogre wiki
- explain why it's needed (polygon precision ray casting -> half cubes, without it only bounding boxes are checked and free space is impenetrable)

3.10.6 SelectionBox

- explain its purpose and how it works
- talk about single/area selection and multi selection using shift

3.10.7 Conditions

- explain their purpose, how they are used in querying and effect application
- mention that it's a good way for extensibility as new can be easily implemented
- mention that they are often used as template parameters and thus need to conform an interface (but not by inheritance)
- talk about the interface
- maybe demonstrate on an example

3.11 Pathfinding

- recapitulate from analysis that a modifiable function was used that allows different algorithms, heuristics and path types
- mentions its parameters (destruction, add path are primary!)
- mention how destruction works
- mentions why path addition is optional (for checks)

3.11.1 Grid

- explain its purpose and implementation
- mention that it only provides a set of grid related operations on a set of IDs it's provided (and assumes those are grid)
- mention random node placement and entity distribution
- mention free node list for ease (and speed) of access
- explain graph creation and linking
- explain general usage

3.11.2 Algorithms

- explain how they should be implemented (what functions, as they are used as template parameters)
- mention that A* is the only currently implemented
- mention portal implementation
- mention difference in complexity (component lookups)
- mention any other differences from a general A*

3.11.3 Heuristics

- explain how they are used and why they are inheriting a base class while other functors are not (state needed for some -i run away heuristic)
- maybe demonstrate on an example as the code is small?

3.11.4 Path Types

- explain what these things are and when they are used to check if the algorithm should stop
- explain their effect on the A* algorithm

3.12 Serialization

- talk about how easy component serialization is and how to create template specializations for new components
- show an example?
- explain the whole saving process, ents to be destroyed, unlocks, player, grid etc.
- explain the loading process

3.13 Player

- just say that it is used as a resource bank, keeping track of gold, mana, units etc
- mention that it also holds the starting unlocks used for new games (saved there during the game initialisation)

3.14 Singleton Design Pattern

- explain what this stuff is, its pros & cons
- mention that the main reason for its use in this game is having one instance, not that much global access

3.14.1 Script

- mention how two Lua virtual machines cannot communicate and data are bound to C++ (via EntitySystem) so it's completely unnecessary to have more than one Scripting engine

3.14.2 GUI

- why would we want to have two GUIs?
- mention that here is the global access very good for testing?

3.14.3 Player

- why would we want two sets of player resources?
- serialization preserves them and shown is only the main set (also used is only the main one)

3.14.4 Grid

- explains how it only operates on a set of given IDs that are made during graph creation (+ mentions it would be easy to implement their switch) so there does not need to be more than one pathfinding grid
- also mention that due to the nature of the game levels, it would be nonsense to have two grids

3.14.5 EntitySystem

- explain how even though it's getting passed around a lot (mainly Helpers), there could be a reason to use more than one EntitySystem instance (like a backup for example) and thus the Singleton pattern might not be the best choice
- maybe compare it to the 4 previous classes

3.15 Scripting

- talk about the API, possibly mention some API reference text file that I should make as an attachment

3.15.1 Initialisation

- talk about the config.lua & init.lua duo
- explain different options in config
- explain how scripts packs are added to init
- talk about the core.lua file of each script pack and what it should do (probably with an example)
- explain how to add a new script directory
- explain how to override some behaviour
- explain the new level callback and for what it can be used (+ the return value meaning)

3.15.2 Entity Representation

- explain the structure of a script representing an entity (with a small example)

3.15.3 Blueprints

- explain what is a blueprint, how to make one and how to use it
- explain why it's a table and not just a function and why the functions in different blueprint should have different names

3.15.4 Research

- explain how to add new unblocks and modify existing ones

3.15.5 Spells

- explain the structure of a spell table and how to make new spell
- mention again the spell types and the difference they make in the spell table function implementations
- mention that spells can be made during run time for testing purposes

3.15.6 Creating a Mod (? Modification)

- provide a simple tutorial on how to make a new mod
- possibly the tower defense one?

3.16 Extending the Game

- small sequence of tasks that are needed for a new feature addition (that is component, system etc.)
- idea: show how a TargetedComponent would be implemented, allowing handlers for targeting/untargeting and how this could be used to create a chess mod

4. User's Documentation

4.1 wubwubwub

Conclusion

Bibliography

J. Anděl. *Základy matematické statistiky*. Druhé opravené vydání. Matfyzpress, Praha, 2007. ISBN 80-7378-001-1.

List of Figures

List of Tables

List of Abbreviations

Attachments