NTNU

Det skapende universitet

Department of
Computer and Information Science

A Component based Game Architecture for Unknown Horizons

Thomas Kinnen

TDT4570 - Game Technology, Specialization Project

Table of Content

1 Introduction

1.1 Motivation

1.2 Problem Statement

1.3 Project Context

1.4 Readers Guide

Probably not... [1] [2] [3] [4]

2 Research Methods and Questions

In this section we present our research questions and methods used in this work.

2.1 Research Questions

We work on a set of four main research questions:

- RQ1: Which architecture is used to describe objects in-game?
- RQ2: How are new objects added to the game?
- RQ3: Can existing objects easily be modified?
- RQ4: Are tools available to help with adding/modifing objects?

[1] Unknown Horizons website: `http://www.unknown-horizons.org`
[2] Battle of Wesnoth website: `http://www.wesnoth.org`
[3] 0 A.D. website: `http://wildfiregames.com/0ad/`
[4] Glest website: `http://glest.org/en/index.php`

2.2 RQ1

Which architecture is used to describe objects in-game?

The goal of this question is to find out if the game uses an inheritance based approach, a component based approach or some other design to describe objects in game.

2.3 RQ2

How are new objects added to the game?

With this question we want to find out if many changes have to be made to the code to add new objects. We also want to know if the objects are data or code driven. If they are data driven, we research which technology is used. Our goal is to find the easiest methode of adding objects to the game.

2.4 RQ3

Can existing objects easily be modified?

Our goal is to assess if existing objects are easily changable or if code has to be changed to modify them. We research what the possible implications of changing objects are.

2.5 RQ4

Are tools available to help with adding/modifing objects?

As creating game content is usually done by non programmers, we want to assess how easy it is for them to add content to the game and if there are tools to support them in the progress.

2.6 Research method

The first part of this work is four case studies in which we research four existing open source games. All games are mainly real-time strategy games, so their implementations face similar problems and are comparable to some degree. We will then use the gained knowledge to improve the handling of objects in *Unknown Horizons* by designing and implementing a system combining the best practices we found in the case studies. Literature research is not a big part of this project, as there is almost none available on the topic of game architectures, besides from massiv multiplayer online role playing games.

3 State-Of-The-Art

3.1 Related Work

3.2 Literature

4 Own Contribution

In this section we present four different case studies to answer our research questions. We begin with presenting *Unknown Horizons* as it is the project which focus our efforts of improvement on. It is followed by *Battle for Wesnoth*, *Mega Glest* and *0 A.D.*. The results are then evaluated and transferred to *Unknown Horizons*.

4.1 Unknown Horizons

Unknown Horizons as described on the project website:

> Unknown Horizons is a 2D realtime strategy simulation with an emphasis on economy and city building. Expand your small settlement to a strong and wealthy colony, collect taxes and supply your inhabitants with valuable goods. Increase your power with a well balanced economy and with strategic trade and diplomacy.

RQ1

Unknown Horizons uses a largely inheritance based approach to describe ingame objects. As the game is programmed using the Python[5] programming language it is possible to use multiple inheritance. The project makes great use of this ability, resulting in great inheritance trees. To illustrate this we have generated an inheritance diagramm for the *Settler* class in Figure 1. The tree consists of 16 classes including many cases of multiple inheritance.

Experience in working on this project has shown that making changes to any of the classes included in this tree is often a very big task and comes with a great risk of introducing bugs into the code. It is also very difficult or even impossible to write unit tests for these classes, as they are so dependent on each other and the game core, that it is almost impossible to create the needed environment synthetically.

Settler Explained The *Settler* class is comprised of 4 basic classes: *BasicBuilding*, *SelectableBuilding*, *BuildableSingle* and *CollectingProducerBuilding*. This is how most buildings in *Unknown Horizons* are constructed.

BasicBuilding is a base class for every building, it loads graphics and provides basic information like the name, position, owner and functionality for running costs and level upgrades.

[5]Python website: `http://www.python.org`

SelectableBuilding is a decorating class, that implements functions for selecting the building ingame. It manages showing ingame menus and outlines. If a building is not supposed to be selectable, this class should not be inherited.

BuildableSingle is a decorating class which is used when building new buildings. It tells the game that it can only be built as single instance, so there is no building of multiple instances at once. For this purpose the code provides the *BuildableLine*, *BuildableRect*, etc. classes which can be used if needed.

CollectingProducerBuilding is a collectiv class to make the Settler have collecting units which pick up resources for usage and then produce something from it. This is easier to demonstrate on a *LumberJack* for example, he picks up trees and produces planks from it. The Settler consumes resources (food, textiles, etc.) and in turn produces the abstract resource happiness.

Datadriven? Unknown Horizons uses a SQLite[6] to save parts of the object's attributes. For example the size, health and name are saved in the database. This is necessary to make the highler level classes in the architecture reusable for subclasses. All buildings have a size, but it may be different from building type to building type. It is saved to an external file to make it easily editable by non programmers.

In summary we can say that the objects are partly datadriven, but usually it is not possilbe to add new buildings without writing new code.

RQ2

In order to add a new building to Unknown Horizons one has to look at the characteristics the building should have and then find the appropriate classes from the Unknown Horizons building classes collection. Those can then be combined to form new buildings.

For example to create a settlement wall one could use the classes *BuildableLine* and *BasicBuilding*. This is a very simple example which does not need to inherit many classes, as its functionality is very limited. All attributes of this building can then be added in the database by using any SQLite database manager.

RQ3

Modifying existing ingame objects in Unknown Horizons can be easy and very difficult. This depends on the degree of change that is to be made. If only basic attributes like health, production time or similar are to be changed, then it can easily be done by someone who knows their way around the database. If however new functionality is required, for example an building which previously did not collect resources needs to collect resources, a change in the games code is most certainly required. Again sometimes if the functionaliy exists, this can be easy by just adding another class to the hierarchy of the building or it can be very difficult if new functionaliy in the existing classes is required.

[6] SQLite website: `http://www.sqlite.org/`

A good example for this is the boatbuilder, which is mainly a *CollectingBuilding* which produces units instead of resources. The building has been implemented for over a year now and the team is still not certain if it works bugfree or not, as it required huge modifications to the production classes to be able to produce units instead of resources.

4.2 Battle for Wesnoth

4.3 Mega Glest

4.4 0 A.D.

4.5 Evaluation

4.6 Transferring the Results to Unknown Horizons

Design

Implementation

5 Evaluation

5.1 Project

5.2 Results

5.3 Methods

6 Conclusion and Future Work

6.1 Conclusion

6.2 Future Work

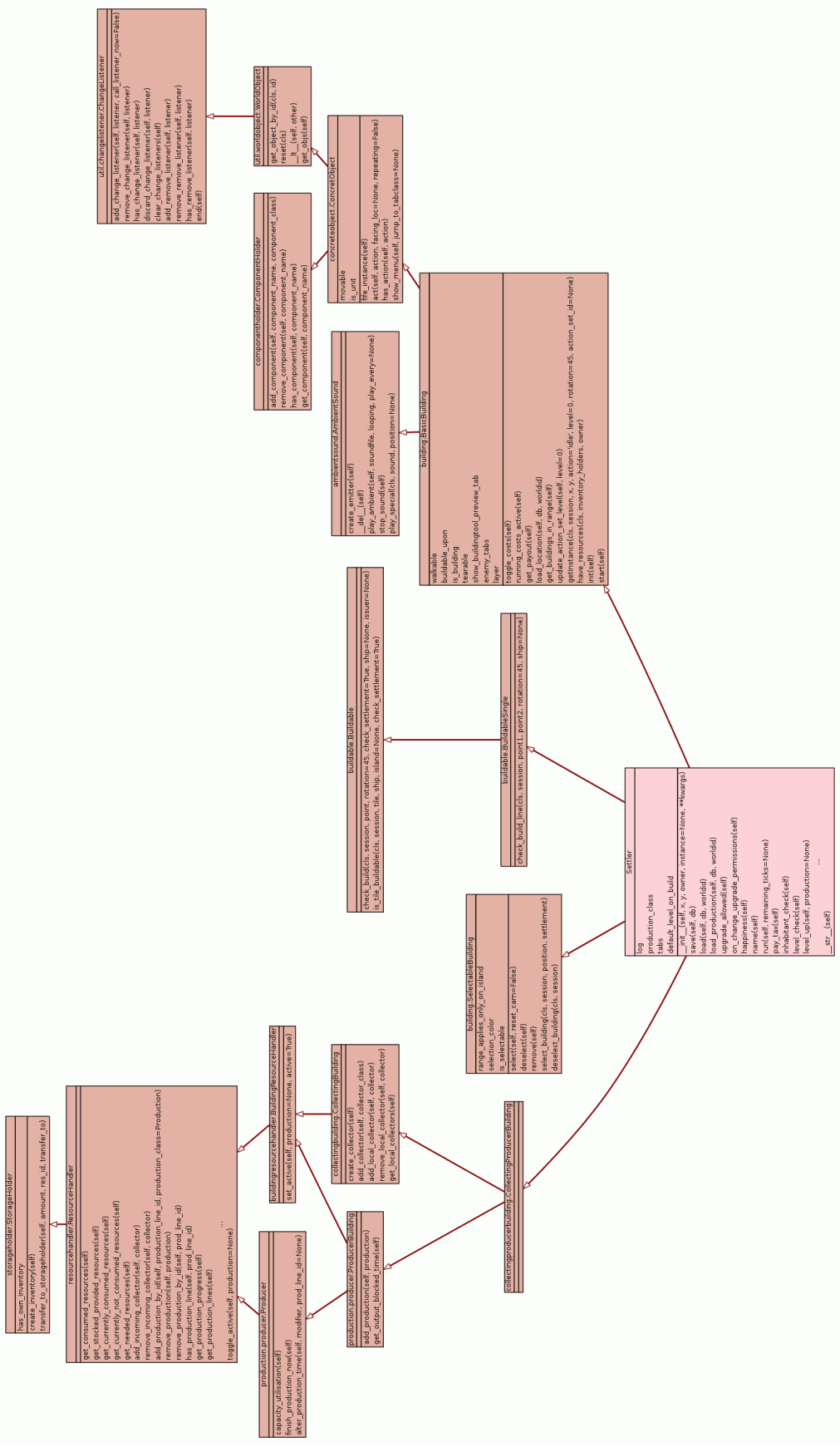

Figure 1: Inheritance tree for the *Settler* class in Unknown Horizons