COP 4331

Object-Oriented Programming

Spring, 2017

Project: Iteration 1

Project Overview: Iteration 1

You will produce a TBS (Turn-Based Strategy) game. Our goal for this first iteration is to develop a solid engine upon which an entertaining game can be built later. Fancy graphics and a storyline are the very last thing you should spend time on.

Logical Game Elements

The model for this iteration shall incorporate several logical game elements--units, structures, items, and maps--which are described below.

Units

An *unit* is a mobile thing which has a specific location and the ability to relocate itself. Units are characterized by stats. This iteration shall define the following entities:

- *Explorer* - explores terrain; terrain must be explored before its resource levels are known
- *Colonist* - required to establish a base
- *Melee unit* - fighter
- *Ranged unit* - ranged fighter

Stats

Stats, short for statistics, are a set of quantified characterizations of an unit's abilities and status. This iteration shall define the following stats:

- *Offensive damage* - damage dealt when attacking

- *Defensive damage* - damage dealt when fending off an attack
- *Armor* - absorbs a fixed amount of damage
- *Movement* - the maximum distance a unit may move in one tick
- *Health* - when reaches zero, unit is destroyed
- *Upkeep* - resources required to keep unit at full health; lack of required resources will adversely affect the health of a unit

Navigation and Interaction

Units may move—when not blocked (*e.g.,* by enemy units) or otherwise constrained—in any of 8 directions: N, NE, E, SE, S, SW, W, and NW. The unit shall be assumed to face the direction of its last (attempted) movement. Interaction shall be accomplished by an (attempt) to move into the tile containing the interactive item.

Structures

A *structure* is an immobile thing which has a specific location, stats, and purpose. This iteration shall define the following entities:

- *Base* - manufactures and heals damaged units
  - *Offensive damage* - damage dealt when attacking
  - *Defensive damage* - damage dealt when fending off an attack
  - *Armor* - absorbs a fixed amount of damage
  - *Production rates* - the number of turns required to produce a unit of a specific type. (Note: time to heal a damaged unit is computed as % damage * production rate.)
  - *Health* - when reaches zero, the base is destroyed
  - *Upkeep* - resources required to keep structure at full health; lack of required resources will adversely affect the health of a structure

When all of a player's structure have been destroyed the game is over.

Items

An *item* is an immobile thing which has a specific location. This iteration shall define the following items:

- *One-shot Item* - activated and removed from map when touched by a Unit
- *Obstacle* - makes tile permanently impassable

Maps

Maps consist of a rectangular grid of locations. Discrete locations in the game world are geometrically square. Each location has a terrain type, may contain entities and items, and may be associated with an area-effect.

Terrain

Terrains indicate the physical characteristics of the "landscape." This iteration shall define the at least three terrain types, one of which must be impassable by the starting units.

Resources

There shall be at least three different kinds of resources necessary to sustain life, build new structures, and support research. Resources shall be distributed unevenly over the map.

Area-Effects

Area-effects are processes that are automatically triggered when an unit enters a place. This iteration shall define the following area-effects:

- *Take damage*
- *Heal damage*
- *Instant death*

Screens

The player's interface shall consist of three screens:

- *Main screen* — for issuing orders, seeing effects
- *Unit overview* — summarizes data about units
- *Structure overview* — summarizes data about structures

Main Screen

The player will primarily interact with the main screen, which have two *viewports*--a categorization of the kind of data that shall be represented--into the game world: an area viewport and status viewport.

Area viewport

An *area* is a representation of the visible subset of the map in which game play is happening. Each of the visible locations is represented by a *tile*, which depicts the terrain type (potentially decorated with a decal), and the unit or item, if any, at that location.

Decal

Decals augment the terrain and primarily serve as eye-candy. They do not intrinsically affect game play--though one may be used to mark a tile to indicate an area-effect, etc. This iteration shall define the following decal types:

- *Red Cross*
- *Skull and Crossbones*

Status viewport

A representation of the player's current structure/unit's stats.

Unit overview

An interactive table which lets the player see at a glance (some scrolling may be required) pertinent stats and missions of each of her units. Armies can be assembled/disbanded from this screen.

Structure overview

An interactive table which lets the player see at a glance (some scrolling may be required) pertinent stats and missions of each of her bases. Cities can be (un)grouped and missions (*e.g.,* production, research, etc.) assigned/modified.

Goals and Anti-goals

Goals

The goals for this iteration include:

- *Each player starts with 2 explorers and a colonist*
- *Players alternating their turns*
- *Unit movement/ missions*
- *Bases orders: produce unit*
- *Bases orders: heal unit*
- *Units interacting with terrain*

ANTI-GOALS

The goals are oriented towards building infrastructure. Therefore, you do not need to focus on gameplay elements such as (but not limited to):

- ~~*Combat*~~
- *Fog of war*
- *Multiple unit or weapon types*
- *Etc.*

Architectural Guidelines

The architecture shall be based upon the MVC (model-view-controller) pattern. This pattern breaks an application into three encapsulated subsystems. The *model* is the heart of the system and contains the data and program logic. A model is independent from and therefore can be presented by multiple views (*e.g.,* consider data in a spreadsheet that can be presented as a table, or a graph, or an XML document). A controller--also independent from the model--is used to modify the model's state (*e.g.,* cutting and pasting text from a document) or manipulate the view (*e.g.,* scrollbars change what is visible, but don't change the fundamental data). A controller could be variously implemented as something that interprets input from a keyboard, mouse, joystick, network, another program, etc. Since the three subsystems are loosely coupled and only communicate through a well defined interface, the advantage of using the MVC is that it allows variations on a subsystem to be interchanged without affecting the correct operation of the other subsystems.

MODEL

The model shall manage and manipulate the logical game elements identified in this document.

View

For the first iteration, you have the option of developing an ASCII text mode or a Java2D graphic view based on the screens previously described. The map shall be larger than the area viewport--which will automatically scroll to keep the player's currently selected unit/structure in the center of the area viewport as s/he moves around.

Controller

The controller shall be based on input from the keyboard and is described in a separate document.