AXIS
USER DOCUMENTATION

Thank you for downloading AXIS. The AXIS Project Group is very happy the project is getting attention from individuals. We hope that people will beta test AXIS milestone releases and perhaps lend time to review and contribute to the design, documentation, and programming code. This document should help you understand how to use the current release of AXIS, contains an overview of the project, what works, what doesn't, and what plans for the future might be.

Table of Contents

OVERVIEW

AXIS is a turn based strategy game. It is not a clone, hack, or rip-off of any other game past or present, computer or board, and is a completely original design. Gameplay is based on a randomly generated tile map. The theme is a kind of abstract geometric mix. Many TBS games on the market carry sci-fi, fantasy, or war themes. AXIS is far removed from these and tries to disassociate itself from them. Instead it uses shapes, colors and textures to convey information. The game is won by capturing another side's flag or annihilating all the playing pieces of that player. Possibly other victory conditions can be incorporated as well.

Quick Project History

[as told be the lead designer, leiavoia]

The idea for this game goes back about 5 years or so. It started out as a primitive cards+dice game. After putting together a decent looking mockup, i realized that even though i had good intentions and designed a reasonably balanced game, this game was *booooring*. The constant drudgery of writing and scoring and keeping track of several dozen little paper punchout bits and hoping nobody sneezes was too much. There were too many calculations and minor details. This game was no good in its current form.

I shelved the game for a year or two. It sat in my closet for all that time with the little pieces and boards and cards packed nicely away. About this time, i started getting more involved with programming and computers and also started following the development process of *Master of Orion III*. I was very interested in certain design aspects of the game as well as learning about artificial intelligence and game design with a programmer's viewpoint. I was inspired to take the ol' cards+dice game i made once upon a time and do something new with it: turn it into a PC game with a heavy AI slant

I quickly learned that programming did not come naturally to me. But i still found the design and the artificial intelligence to be fascinating. So now, i have this wonderful idea for a game and *some* programming experience, but mostly in scripting languages like Perl and PHP and just enough object orientation to be slightly dangerous, or at least know what I'm talking about.

So now i am opening the project up to a wider audience. There are some very good concepts here, but we need the help, both in programming and in design. I want this project to be open for discussion and also open sourced, mainly because i am starting the project for fun and not for profit. That and h'm also a Linux / open-source fan and i don't want another game that only runs in Windows!

Design Goals

The goal is to create a strategy game that is simple but deep. Easier said than done. By watching Master of Orion III's example, which tried to accomplish that same thing, i have learned that bigger is not always better. There must be ways to add more fun and strategic depth without making a game that is overly complex. Therefore, the strategy in AXIS should come from carefully placed game elements and a push and pull between opposing ideas (offense verses defense, for example), not from a plethora of details and statistics.

Another design goal is to create a game in which you are the strategist not the tactician. You should not have to make every little move. It would be more interesting to give each unit or group a mind of it's own and a set of behaviors to work with, then try to work with them in a more hands off fashion. This concept is interesting but is difficult to implement effectively. However, dealing with the unit's intelligence is part of the strategy of winning.

Designing an AI intensive game not only adds a very different approach to the Turn Based Strategy game genre, but it also allows the computer to be able to play against you. Many open-source games in development have only a multiplayer feature because there is no AI to play against. If each unit and group in AXIS is programmed with its own little brain and makes its own decisions, we are already closer to having a game that works as well in single player as it does in multiplayer.

Another problem that can take down a good strategy game is time. Some games take an eternity to play. The goal for AXIS is to have a game that is relatively engaging timewise but that does not take a week to finish. However, AXIS games should be scalable; those who want a fire-fight kind of game can play in a crowded playing field and a small map, those who want a lengthy campaign can have it their way as well with a very large map. Startup options should therefore be plentiful. This adds replay value. However, we want to avoid feature creep and bomb the player with too many options. The design philosophy here is: lots of startup options, but only where they make a difference. Lots of little options are interesting, but may have no perceivable effect on gameplay and waste a lot of time and attention in the meantime.

Inspirations

These ideas did not come from out of nowhere. The original concept (in the cards+dice version) was meant to be a modified and more complex form of the board game Stratego. The AI-control ideas come from several places, but most notably the program-and-watch play of Maxis' RoboSport and some macromanagement concepts in the original Master of Orion III design.

GAMEPLAY

Grouping

The base unit of play in AXIS is the *shape*. Shapes come in many fun flavors including Tetrahedrons, Tori, Cubes, Spheres, etc. Each shape has a variety of Final Fantasy style combat statistics which may include force (offensive power), resistance (defense), opacity (evade), acuity (accuracy), speed, and many others. In addition, different kinds of shapes carry different strengths and weaknesses. Most shapes carry special skills which can come in handy in and out of combat.

The next level up from a shape is the *pod*. A pod is simply a group of shapes (think "task force"). Pods are what you see on the overall map. Once shapes are assembled into pods, they can move about on the map tiles. When conflict occurs, it happens between two pods and the shapes that make up the pods do battle in some graphically abstract fashion.

The final level of grouping is the *axis* from which this game gets its name. The axis is the "empire" you control. There can be many axes on the map. The current idea is to have 2-4 axes on the map, depending on startup configurations. Possibly even more than that could be added, but the current limit is 4 players for computational reasons.

Note: For this release, the emphasis is on the pod level. Therefore, you will not see much of shapes in this release. Each pod is given a single shape (since it requires at least one to be in existence)

The Map

The map is square tile based. Something that will surely increase fun and replay value is a randomly generated map. AXIS' random map generator is called “BlotchMaker” and has a large suite of tools and variables for creating random maps. Map creation is it's own art and you can spend a long time just learning how to coax BlotchMaker into drawing some really interesting playing fields. If you come up with a really great set of configurations, be sure to save them and post or trade the config file.

v0.1 Note: Blotchmaker has many features in this release, but the Back, Randomize, and Save and Load Map buttons currently have no function. You can save and load configurations however; these are loaded into and from a file called userdata/last.bmc via buttons on the BlotchMaker screen.

Each map tile has some associated statistics. Notably, there are obstacle and road quality. The obstacle explains itself. Each tile can either be EMPTY, BLOCKED, or any one of several others. BLOCK is completely impenetrable. EMPTY can be moved across by all pods. The others are conditional: you can move across them if you have the ability to do so. Otherwise, it acts as a BLOCK. Road quality determines how “rough” the tile is to travel across. A pod traveling across rough road (~10) will not get far as it takes more effort to travel across it. Pods moving across compacted and well traveled roads (~1) will go significantly farther.

Movement and Missions

Each axis (player) starts off with a handful of shapes in a reserve. These shapes can then be assembled into pods based on what function or mission the player wishes that pod to accomplish. For instance: an attack pod should be high in average shape attack statistics. A scout pod should have high overall visibility and shapes with recon specials. Certain shapes lend themselves to certain tasks, but pod design is up to the player to experiment with what works and what does not.

v0.1 Note: There is no pod reserve in this version. Pods are all assigned one shape as contents

(which you will never see). Pod placement and removal is facilitated through the [R] button on the navigation panel for manual pod placement and deletion wherever you wish.

Once assembled, pods are placed into the axis' *home base*. This is also where the axis' flag is kept. Once placed, the pod is then given a player defined objective which can be one of many or queued so that once a pod finishes one objective it can start on the next automatically. Pod missions are given by the player, however, the way the pod accomplishes this mission is left up to the pod itself. This is what makes AXIS somewhat unique: the heavy AI emphasis on several levels. Pod missions may be direct like "Move to square [*point*]" or vague like "Patrol Area". But after having received orders from you, the pod then decides how best to achieve its mission. If you say "Move to tile [x,y]" it decides how it should get there on its own. If you say "Patrol Area" it scouts its own territory and decides who and how to attack if needed. A short list of pod missions includes but is not limited to: Move, Attack, Follow, Scout, Capture Flag, Defend, Halt, Retreat, Patrol, Return, and Harvest.

Movement takes place in turns. Once all pods needing orders from all axis' have been instructed and the Turn button has been pressed, each pod moves across the map. Pods with higher average initiative values go first. Each pod can move several squares depending on several pod and shape statistics.

v0.1 Note: pods move in the order they are placed for this version. The initiative system is present but not implemented in the turn system.

USER INTERFACE

The Start Screen

The start screen is the first screen you will see when starting a new game. It pretty much explains itself, so we won't go into too much detail here.

v0.1 note: Only the START NEW button works for now. It will forward you to the map generation screen.

Random Map Generation (BlotchMaker)

The BlotchMaker ("BM") screen is where you configure the map for the game. As you can see, it has more options than anyone really needs (ah, the bane of freelance software production!). However, playing with blotchmaker can be a lot of fun, so let's learn what all these buttons do:

The options are split into three groups: 1) the button panel (towards the bottom), 2) the Global Settings area, and 3) the Slot Options.

Global settings obviously effect the generator as a whole. The generator works off a set of 5 "slots", each containing a set of variables. When the generator runs, it follows a basic sequence of events:

1. BM rolls a virtual die to choose which slot of variables it will use to make the next "blotch".
2. BM lays a tile randomly somewhere on the map surface. It then picks a direction for each subsequent tile to be placed based on a set of "turn persuasions" and places all remaining tiles it is alloted for that one blotch.
3. BM will draw another blotch, do another die roll, and so on until it has created the specified number of blotches.
4. The blotched map is verified as playable .

Global Options

Map Size

Is the size of the map in tiles, square. Changing the map size will clear the current map automatically.

v0.1 Note: Yes, there are plans to have independent X,Y sizes in the future. It is possible in code, but is not available in the BlotchMaker interface at present.

Number of Blotches

This controls how many total blotches blotchmaker will place on the map. It does not determine of what kind the blotches are (what slot of variables they are drawn from).

Slot Lottery Balls

Each of the 5 variables slots gets a number of lottery balls assigned to it for a chance to be drawn. When the drawing for a blotch occurs, the program will put all the lottery balls together and draw one out to choose a slot to work with. Lottery balls do not deplete. That is, if Slot 0 was set with 10 balls and was chosen for the first blotch, it will not have 9 on the next round. Lottery balls are just a way to represent ratios and probabilities.

Map Initializer

This is the type of obstacle that the map will initialize to. By default, it is set to EMPTY tiles.

v0.1 Note: There are several kinds of other obstacles including water, pits, barriers, and more that are not included in this version.

Slot Options

Slot Number

The number of the slot the screen is displaying. There are 5 slots. If you change the slot number, the rest of the slot options will change according to what variables the slot contains.

Obstacle Type

The type of obstacle the blotches made will consist of.

Turn Chances

As with the Slot Lottery Balls in the Global Settings, each relative direction (forward, left, right, and back) has a number of lotto balls associated with them for drawing a random direction. If you had balls assigned like this: Forward=3, Left=1, Right=0, Backward=0, then you would have a 75% chance on going forward on the next tile placed in the blotch and a 25% chance of turning left. Obviously, this option is one the biggest characteristics of the blotch.

Tiles Per Blotch

How many tiles the blotch will consist of

Mode Chances

There are two modes (and some more on the design table) for placing tiles. “Chaotic” places as many tiles as it can, but will quit if it hits a wall. Also, tiles that are placed over other tiles count as tiles placed. “X_OR” is a bit-flipper. If a BLOCK tile is placed, it will EMPTY it and vice-versa. You can configure the lottery balls associated with each mode to mix it up a bit.

Edge Gravity

Edge gravity pulls the initial tile in a blotch towards (or away from) the edge . It does not effect the rest of the tiles placed in the blotch. The higher the gravity, the harder the edge will “pull” the first

tile towards the edge. Likewise, the lower the gravity (from zero), the further the first tile will tend towards the center of the map. Edge gravity is felt stronger towards the wall itself than the center of the map. Therefore, if the first tile happens to be placed in the direct center of the map, you will not see any change whereas a tile placed *near* the edge will probably hug the edge. A value of zero means that the walls neither push nor pull and have no effect.

Brush Type

When a tile in a blotch is placed, it is actually placing a "brush stroke", or a pattern of tiles around a center point. The shape, or brush type, can be changed and has a great impact on the character of the resulting blotch overall.

Brush Step

Determines how far a space (in tiles) separates each brush placement in a blotch. The default is one.

Deterioration %

Each individual tile in a brush placement has a percent chance of "shooting a blank". This chance is rolled for each tile, not for the brush stroke as a whole.

Blank Shot %

The chance for the entire brush stroke to "shoot a blank" as a whole.

Buttons

When the configuration is too your liking, press REGENERATE to recreate the map. You can do this as many times as you wish until you get a map you like. Press DONE when the map is to your satisfaction. This will forward you to the map screen. LOAD and SAVE CONFIG will save the current blotchmaker settings to a single file for later use.

v0.1 Note: There can only be one file at this time. There is no file selector to save multiple configs. Also, the BACK, RANDOMIZE, and MAP saving and loading are not functional in this version.

Verification

All maps must be verified before they "go to press." BlotchMaker will try to trace a path from each home base area to each other homebase area. If it fails on any path, the map is a bad map and not suitable for gameplay.

Pathfinding is a CPU intensive function. If the map is very large, if there are many players, or if the BM configuration is a certain way, the map verification process can take a long time even on newer computers. To limit this, try not to create maps with configurations that make patterns with lots of open spaces but no traceable routes. For instance, a square open map with a solid line down the middle will max out the pathfinding time since it will search every open tile looking for a route across, but you will *still* get a bad map.

Blotchmaker will make three internal attempts to blotch a good map without telling you. If it blotches a good verifiable map, you need not worry. If it fails 3 consecutive times, it will let you know it could not create a good map with the current settings. If you get this message a lot, try a different set of configurations that opens up more area for the pathfinder to connect home bases.

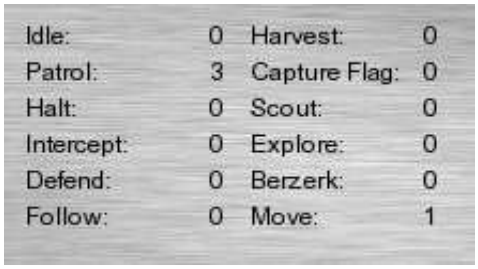
The Map Screen

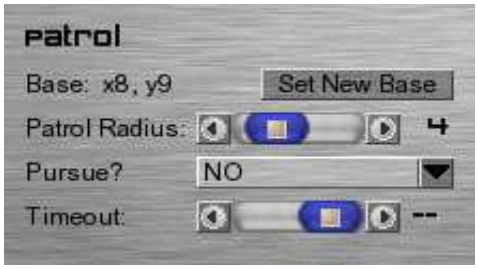
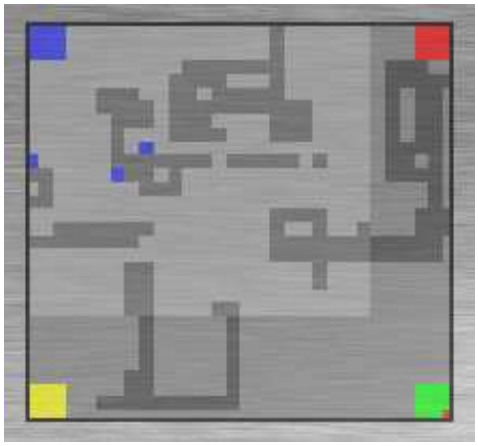
This is where most of your time will be spent. The map screen has several major elements. The map itself takes up the most room. It is scrollable via the scrollbars. A status bar at the top will keep you informed of what the computer is doing. The NEXT TURN button is in the upper left. Clicking it will move each pod or execute it's assigned mission (if any) and will resolve any combat that needs to take place.

The largest element is “The Panel” on the left side. It is the main interface to the game. It has (or will have) everything from game options to pod overviews to reserve deployment. It is a large enough element to warrant discussion of its own:

The Panel

The panel is the main interface to the game. It “rotates” to display many different panel screens. Each different screen has a set of stackable panel modules. While some modules are seen on many screens, others are unique to a specific panel screen. Each panel module is presented here with a description:

	This is the Panel's navigation bar. You can use it to switch between the Panel's different subscreens. Here, M = “Map Screen”, P = “Pod View”, R = “Reserve”, and D = “Demo”.
	The selected tile module. Clicking on a map tile will load a tile into the module here. The tile itself is displayed along with it's coordinates and it's road quality. If the tile is occupied, it will also show the occupant, it's name, and it's current mission
	Mission Counter. This shows how many pods are on the different missions in your axis.
	This is the mission selection list. You may add and remove missions from the pod's mission queue. Click the mission slot up or down to highlight the slot in the queue you wish to work with, and either add a new mission in that slot or remove the mission currently in that slot. The list also contains a few brief details about the missions in queue

	This is the mission details module. It lets you configure the parameters for the mission currently selected in the selection list above. The interface for this module changes depending on what mission is selected. Some missions have no configurable option and the number of options varies with each different mission type.
	This is the pod's general mission settings: Avoidance, and Engagement. Avoidance rates how strongly a pod will avoid enemies. 0 = not at all, 9 = at all costs. Engagement determines what other pods this pod will try to attack, if any.
	This is the minimap. It displays the entire map in a small concentrated area. Light grey areas are open space. Dark grey areas are blocked. Colored areas (including blank and white) are either pods on the map or are home bases.
Note: For the duration of the development process, there will usually be other panel modes and modules to facilitate debugging and development functions. These will not be documented as they will probably be in a constant state of flux. Hopefully, you can figure out what they do on your own.	

ABOUT VERSION 0.1

Version 0.1, also known as “Advanced Football”, is a pre-alpha testbed for future development. It was not meant to be playable in it's current state a*s a game*. The basic requirements for this version were a good backend set of data structures for pods and map generation, and a usable graphical interface. The other requirements were movable pods that could be assigned and execute mission orders. These things have been achieved!

Since we now have the framework for future development, progress should move even faster now than it has. Many of the problems of getting the game to work have already been tackled, and we only fall for the same bug tricks once.

Version 0.2, the next milestone, calls for an expanded set of missions and more hands-on with shapes. The shape generation engine should be fully in place by 0.2. Other major new features include map resources, including axis flags, that can be located and collected by pods. Version 0.2 should really start to shape the project, but still leave it unplayable as a *game*.

Hopefully, you will help us work out the bugs and give us ideas and suggestions for future development. We'll strive to make AXIS a very solid game, but it takes more pairs of eyes than one.

Please feel free to contribute thoughts, ideas, praise, suggestions, critique, or whatever on the AXIS website.

FOR MORE INFORMATION:

Visit the official AXIS website here:

http://www.project-axis.net/

The CVS code repository is located at Savannah:

https://savannah.nongnu.org/cvs/?group=projectaxis

You can also send email directly (although not recommended) to:

info [at] project-axis [dot] net