Solving Axiom

Jonathan Martin
Outline

Birth of My Project Slide 3
Cool Axiom pictures Slide 4-9
Basic Rules Slide 10-11
Object of the Game Slide 12
Quick Demonstration Slide 13 (link)
Axiom's Creator Slide 14
Perspective Slide 15
The Task Slide 16-17
The Goal Slide 18
Steps to Achieve Goal Slide 19-26
Fruits of Our Labor Slide 27-28
What I Learned from this Project Slide 29
Works Cited Slide 30
Birth of My Project...

- Learning a new programming language and interpreting brain waves sounded like more difficulty than it was worth...

- Solving a game = WAY COOL!

**To "solve" a game refers to the act of determining whether a game, given perfect play by both players, has a predisposed winner.**
The Game
Sceptre and Cube
Starting Position

Black Axiom pieces

Black & White start position
Possible Ending Position
Position
Positions are indicated by three numbers. \((x,y,z)\)

The first two numbers \((x,y)\) show the position of the cube on the grid. The last number \((z)\) shows the height above the sheet of paper.

The diagram below, illustrates the 3D coordinates.

Directions
Spikes and sceptres can point in any of the 6 directions. These are shown, along with the abbreviations, in the table below:

<table>
<thead>
<tr>
<th>Direction</th>
<th>Abbreviations</th>
</tr>
</thead>
<tbody>
<tr>
<td>up away from the paper</td>
<td>(z^+)</td>
</tr>
<tr>
<td>down towards the paper</td>
<td>(z^-)</td>
</tr>
<tr>
<td>to the right of the paper</td>
<td>(x^+)</td>
</tr>
<tr>
<td>to the left of the paper</td>
<td>(x^-)</td>
</tr>
<tr>
<td>to the top of the paper</td>
<td>(y^+)</td>
</tr>
<tr>
<td>to the bottom of the paper</td>
<td>(y^-)</td>
</tr>
</tbody>
</table>
Starting Positions - per notation system

**Movement of Cubes**
C (old x, old y, old z) (new x, new y, new z) dd
Moving a cube is described by writing down the old position of the cube, then the new position of the cube. You don’t need to note the old position of the spikes, because only one cube may occupy that space.

**Movement of Sceptre**
S (old x, old y, old z) old d (new x, new y, new z) new d
Moving a sceptre is described by writing down the old position of the sceptre, then the new position. Note that you must write down both the old direction and the new direction. This is because you can have more than one sceptre on the same cube.

**The Axiom start position using this notation system.**

<table>
<thead>
<tr>
<th>Black</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C (-1,0,1)</td>
<td>x- z+</td>
<td>C (0,0,1)</td>
<td>x- y-</td>
</tr>
<tr>
<td>C (-1,0,2)</td>
<td>x+</td>
<td>C (0,0,2)</td>
<td>z-</td>
</tr>
<tr>
<td>S (-1,0,2)</td>
<td>z+</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>White</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C (-1,1,1)</td>
<td>x+ z+</td>
<td>C (0,1,1)</td>
<td>x+ y+</td>
</tr>
<tr>
<td>C (-1,1,2)</td>
<td>x-</td>
<td>C (0,1,2)</td>
<td>z-</td>
</tr>
<tr>
<td>S (-1,1,2)</td>
<td>z+</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C (1,0,1) x- z+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C (1,0,2) x+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S (1,0,2) z+</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Basic Rules

- A player may move either a sceptre or a cube during his turn.
- A player must move during his turn.
- Sceptres may be moved diagonally on the flats, and may move through different planes while traveling horizontally.
- A sceptre may be moved as many spaces as desired during a turn until it reaches a physical obstruction (a dome, another sceptre, or the ground).
Basic Rules

- Cubes may be moved in any fashion as desired as long as it interlocks with another cube. However, the game does not permit a cube to "float," meaning there must be either another cube or the ground directly underneath every cube.

- If a sceptre currently occupies a cube of the opposing color, if, when the sceptre is moved somewhere else, the previously occupied cube becomes free, the newly freed cube is then removed from play.
Object of the Game

To win a game of Axiom:

- One must move their sceptre to a cube that is occupied by an opponent's sceptre.

- If the game comes to a point where, during a certain player's turn, they either have no moves or cannot move without putting themselves into "checkmate," then that player loses.
Quick Demonstration...

http://www.axiomgame.com/animation01.html
Micheal Seal

1981 - 1985: Brighton University / School of Architecture - B.A. Degree in Architectural Design
1998 - 2006: Director of Lumicube Ltd. - Lighting design, manufacture and consultancy, Brighton UK. Producing Perspex lighting & illuminated furniture commission and to contract.

Accomplished 3-D designer and inventor of two unique boardgames.
Perspective

Chess
Average branching factor ~ 35 moves
Average depth of a game ~ 100 turns
\[35^{100} = 10^{154}\] possible nodes

Tic-Tac-Toe
~5 legal moves
Total of 9 moves
\[5^{9} = 1,953,125\] nodes

Axiom
Average branching factor: I estimate
(extremely roughly) 20 moves
Average depth of game: I estimate
(extremely roughly) 20 turns
\[20^{20} = 10^{25}\] possible nodes
The Task

As Dr. Goadrich and I found out during our very first meeting, representing Axiom would be much more difficult than we had initially thought.

Though the game play is straightforward enough, having to generate the lists of all possible moves from any given game state proved to be a formidable task.
The Task

- Sceptres move multiple directions
- Sceptres move through 3-D space
- The ability of the cubes to move means that the playing surface is ever changing.
- Finally, due to the ability of the playing surface to be altered, we realized that the potential for an infinite number of possible unique game states exists.

Nevertheless, we decided to cross that bridge when we came to it and proceed with our task.
The Goal

Our goal for this project was to:

1. Be able to accurately represent the game play of Axiom, using the programming language Java, virtually within the computer.
2. Be able to play the game of Axiom with all the rules intact by passing in Strings of coordinates of where a certain piece is located and where we would like to move it to.
3. Use MINIMAX and AlphaBeta pruning algorithms to explore Axiom's full set of unique game states and ultimately "solve" Axiom, if possible.
Steps to Achieve Goal

1. We created a Cube object

2. We used a Hashmap that stored items (type Cube), which corresponded with keys (type String), called "board."

3. We "named" each face of the cube according to which dimension it faced on a 3-D grid
4. We assigned each possible face condition to a number; we then stored each face's number in an array of integers. The array's indices correspond to a direction on the 3-D grid.

"if (faces[i] == DOME && board.get(neighbor).getFace(XDOWN) == EMPTY)"

XDOWN represents the index "faces[1]," while DOME and EMPTY represent conditions of the face being iterated through and the XDOWN face of the neighbor cube, respectively.
Steps to Achieve Goal

5. A Cube is represented by:
   1. An X coordinate
   2. A Y coordinate
   3. A Z coordinate
   4. The direction of its first Dome
   5. The direction of its second Dome (if it has one)
   6. And its color.

"Cube b1 = new Cube(-1,0,2, Cube.XUP, Cube.NONE, Cube.BLACK);"
Steps to Achieve Goal

6. We chose to represent the sceptres simply as a possible condition of a face; we named these conditions "WHITE," for when a white sceptre occupied the face, and "BLACK," for when a black sceptre occupied the face.

Thus, one's code: "board.get(c).getFace(ZUP)" might return: "WHITE", meaning a white sceptre occupies the top face of Cube c.
Steps to Achieve Goal

7. We based our piece movement off a play-by-email version of Axiom, where the positions of the Cube and Sceptres, along with the location of the destination were represented with the following notation:

"S (-1,0,2) S (0,1,2)z+"

This simply means the player would like to move his Sceptre in the Cube located at (-1,0,2) on the grid to the Cube located at (0,1,2), with the Sceptre pointing upward in the "ZUP" direction.
8. Taking the Strings arranged in the notation you just saw, we parsed through to extract the pieces of information to create new cubes in the desired locations and change the location of the sceptres.

When moving a sceptre, we simply changed the condition of both the currently occupied face and the destination face accordingly.

When moving a cube, we simply pieced together the location and created a new cube in the specified destination, and deleted the cube specified to be moved in the string.
Steps to Achieve Goal

9. After nailing down all the Axiom code, would would then apply the MINIMAX player algorithm (suited for Axiom) and the Alpha-Beta Pruning playing algorithm and analyze the results. We developed an evaluation function that would allow us to compare the probable value of each game state.

My evaluation function is:
(number of free **cubes * 3) + number of **cubes = state score
**only that player's cubes
Steps to Achieve Goal

10. If the vastness of the game tree allows, we might be able to identify every possible move.

At that point, our Pruning Algorithm will not only have cut down on the time cost and the number of nodes searched through, but it will also have determined whether there is a predisposed terminal state for the game; that is, whether either the first player or the second player, provided perfect play, was destined to win from the start.
Fruits of Our Labor

Though I just detailed how our goal would have been achieve, unfortunately I was not able to obtain my goal.

Axiom remains unsolved...for now!

We were, however, able to represent Axiom in its entirely, save a method or two, and test the movement of pieces.

Being able to watch Computer agents make movement choices
Fruits of Our Labor

Playing the game via our code was very exciting.

Honing my skills as a programmer; developing experience and enjoyment.
What I Learned from this Project

I increased my programming skills.

Gained more thorough understanding of processes.

Received practice with developing from the ground, up; as opposed to altering existing code.