CodeRulers: Tutorial

Abram Hindle

Department of Computer Science

University of Victoria

abez@uvic.ca

July 24, 2004

Introduction

- What am I going to cover?
  - Goals of a CodeRuler
  - Pieces
  - Peasants , Castles , Knights
  - Game API
  - Sample Source Code
  - Gotchas

Goals

- What are my goals in CodeRulers? [Li04]
- Use your peasants to claim as much land as possible (and keep it claimed).
- Use your knights to capture as many of your opponent's peasants as possible, stopping them from claiming land.
- Use your knights to combat and capture as many of your opponent's knights as possible. This weakens your opponent's defensive capability.
- Use your knights to attempt capture of the other ruler's castle. Castles are factories for knights and peasants; you cannot create more peasants or knights without them. With multiple castles, you gain the ability to create peasants or knights at a faster rate than your opponents.
- Strategically prevent the capture of your castle.

Pieces

- What pieces are there?
  - Castles - do not move, can be captured and still exist.
  - Knights - move, can be destructively
  - Peasents

Peasents

- What do Peasents do?
  - Capture Land (the bulk of your final points)
    - The more land you have the faster you produce knights and peasents.

Castles

- What do Castles do?
  - Generate Peasents and Knights
  - Capturable (they are the little dot at the center of a castle
  - You can own 0 or more castles but start with one.
  - Only knights can capture a castle.

Castles

- Production Rates: [IBM04]

Land you own	Number of turns to create one peasant or knight
124 or fewer	No creation
125	14
250	12
500	10
1000	8
2000	6
4000	4

Knights

- What can knights do?
  - Knights can capture knights, peasents and castles
  - Knights have 100 hp to begin with and get 20 hp per knight captured.
  - Hitpoints are strength getStrength() will return a knight’s strength

Capturing

- What does capturing do?
  - Capturing is a PC term for killing
  - Knights have hit points often you need to capture a knight for a few rounds before you take them.
  - Once a piece is captured it is removed from the game.
  - A capture against a knight takes away 15 to 30 hp.

Objects

- What are objects in the game
  - Ruler - a Ruler. You are to implement this interface
  - IObject - this is the parent class for any object on the board, knights, castles, peasents.
  - IKnight - Knight Interface
  - IPeasent - Peasent Interface
  - World - The World, this object provides access to all the objects in the world.

Objects

- What should I know about the Ruler object?
  - You should explore what it does before anything. The thing that will waste your time the most is implementing and testing things which are already made for you.
  - Directions: `MOVE_NONE` , `MOVE_N` , `MOVE_NE` , `MOVE_E` , `MOVE_SE` , `MOVE_S` , `MOVE_SW` , `MOVE_W` , `MOVE_NW`
  - Getters: getCastles() , getKnights() , getOwnedLandCount() , getPeasants() , getPoints() , getRulerName() , getSchoolName()

Objects

- What should I know about the Ruler object?
  - Commands: (Note issuing 2 different commands to 1 object causes the object to take the last command (e.g. capture and move, means move)
  - ∗ capture(Knight k,int dir)
    ∗ createKnight(Castle c)
    ∗ createPeasents(Castle c)
    ∗ move(Knight k,int dir)
    ∗ move(Peasent k,int dir)
  - What you are to implement
    ∗ initalize()
    ∗ orderSubjects()

Objects

- What should I know about the World object?
  - The world object contains many useful routines which will speed the development of your CodeRuler
  - Defines: `MAX_TURNS WIDTH HEIGHT`
  - Getters: getCurrentTurn() , getOtherRulers() , getOtherPeasants() , getOtherKnights , getOtherCastles()
  - Map Based Getters: getObjectAt(int x,int y), getLandOwner(int x,int y)
  - Point Calculation: getPositionAfterMove(int currentx,int currenty, int dir)

Objects

- What should I know about the IObject objects?
  - getDistanceTo and getDirectionTo are very very useful. Allowing you make quick and fast local optimizations without much coding.
  - public int getId()
  - public int getX()
  - public int getY()
  - public IRuler getRuler()
  - public boolean isAlive()
  - public int getDistanceTo(int x, int y) - Returns the minimum number of squares (units) between this object and the given position. This is also the minimum number of moves that would be required to move to the given position.

  - public int getDirectionTo(int x, int y) - Returns the approximate direction (the closest of the 8 possible move directions) to the given position from this object.

Stuff

- You should be aware:
  - You can’t modify your peasents, knights etc. You can only give them commands.
  - You can’t modify anyone else’s peasents.

Irritating Things

- What is going on!
  - Each object has an ID.
  - Lower ID objects move first.
  - Lower ID objects capture first.
  - I do not know how ID’s are allocated
  - So you can capture into empty space if the other player moves first

Irritating Things

- How do I run my code against my code?
  - Start the server.
  - Submit your code.
  - Stop the server.
  - Find the ”tournament” subdirectory of the server’s working directory.
  - Rename the subdirectory which matches your hostname to something else.
  - Start the server again.
  - Modify MyRuler.getRulerName so you can tell entries apart.
  - Submit your code again.
  - Battle!
  - Taken from http://www.alphaworks.ibm.com/forum/coderuler.nsf/current/F97735525EDA8420D46B0C7D

by MikeG

Strategies

- There are multiple strategies
  - Kill all peasents
  - Capture Castles
  - Focus on Knights
  - Let your castle get capture and recapture it
  - Statistically analyze the movements of the enemy units
  - Use Knights in formations
  - Use random methods
  - Partition strategies between objects.

Required Reading

- Read
  - http://www-106.ibm.com/developerworks/java/library/j-coderuler/ [Li04]
  - Code Ruler's Manual [IBM04]
- Review
  - Source Code (Don't steal ok)

    http://www.cs.ucsd.edu/users/calder/UCSDProgramContest/results/

Example Code

- Disclaimer: This is crap. It works ok but seriously make your own. There is some bad coding here and you shouldn’t use much of it. It provides an example of the problems you get when you don’t use the methods provided to you.

- **MyRuler.java**

Listing 1: MyRuler.java

```
import com.ibm.ruler.*;
import java.util.*;
import java.awt.Point;
/**
 * This is the class that you must implement to enable your ruler within
 * the CodeRuler environment. Adding code to these methods will give your ruler
 * its personality and allow it to compete.
 */
public class MyRuler extends Ruler {
        /* (non-Javadoc)
         * @see com.ibm.ruler.Ruler#getRulerName()
         */
        public String getRulerName() {
                return "Abez’s␣Left␣Iron␣Fist";
        }

        /* (non-Javadoc)
         * @see com.ibm.ruler.Ruler#getSchoolName()
         */
        public String getSchoolName() {
                return "UVic";
        }

        /* (non-Javadoc)
         * @see com.ibm.ruler.Ruler#initialize()
         */
        public void initialize() {
                // put implementation here
        }

        /* (non-Javadoc)
         * @see com.ibm.ruler.Ruler#orderSubjects(int)
         */
        public void orderSubjects(int lastMoveTime) {
                // put implementation here
                int [][] peasantMoveAble = new int[World.HEIGHT][World.WIDTH];
                int [][] knightMoveAble = new int[World.HEIGHT][World.WIDTH];
                int [][] knightCapturable = new int[World.HEIGHT][World.WIDTH];
```

```
for (int y = 0 ; y < World.HEIGHT ; y++) {
        for (int x = 0 ; x < World.WIDTH ; x++) {
                        knightCapturable[y][x] = 0;
                        peasantMoveAble[y][x] = 1;
                        knightMoveAble[y][x] = 1;
                        if (World.getLandOwner(x,y)!=this) {
                                peasantMoveAble[y][x]++;
                                //knightMoveAble[y][x]++;
                        }
                        IObject obj = World.getObjectAt(x,y);
                        if (obj!=null) {
                                peasantMoveAble[y][x]=0;
                                if (obj!=null && obj.getRuler() != t
                                        knightCapturable[y][x]++;
                                }
                        }
        }
}
IPeasant [] peasants = getPeasants();
for (int i = 0 ;  i < peasants.length ; i++) {
                move(peasants[i],getMaxDirection(peasants[i],peasant
}
IKnight [] knights = getKnights();
for (int i = 0 ;  i < knights.length ; i++) {
        boolean  done = false;
        int dir = dirToClosestCapturable(knights[i]);
        Point p = World.getPositionAfterMove(knights[i].getX(),knigh
        int x = knights[i].getX()+k*dirs[dir][1];
        int y = knights[i].getY()+k*dirs[dir][0];
        IObject o = World.getObjectAt(x,y);
        if ( o !=null && o.getRuler() != this ) {
                capture(knights[i],dir);
                done = true;
        }
        if (!done) {
                move(knights[i],dir);
        }
}
ICastle [] castles = getCastles();
for (int i = 0 ;  i < castles.length ; i++) {
        if (castles[i].getRuler() == this) {
                if (Math.random() > .5) {
```

```
                                createPeasants(castles[i]);
                        } else {
                                createKnights(castles[i]);
                        }

                }
        }
}
int getDir(IObject obj, int [] yx) {
        int dist = 100000000;
        int mindir = MOVE_NONE;
        int xo = obj.getX();
        int yo = obj.getY();
        int x = yx[1];
        int y = yx[0];
        int xx;
        int yy;
        for (int i = 0 ; i < dirs.length ; i++) {
                xx = x - xo + dirs[i][1];
                xx *= xx;
                yy = y -yo + dirs[i][0];
```

```
                        yy *= yy;
                        yy = yy + xx;
                        if (dist > yy) {
                                dist = yy;
                                mindir = dirs[i][2];
                        }
                }
                return mindir;
        }
        int dirToClosestCapturable(IObject obj) {
                int dir = MOVE_NONE;
                double dist = 100000000;
                IObject [][]castles = new IObject [][] {
                        World.getOtherCastles(),
                        World.getOtherKnights(),
                        World.getOtherPeasants(),
                };
                for (int i = 0 ; i < castles.length ; i++) {
                        for (int k = 0; k < castles[i].length ; k++) {
                                if (castles[i][k].getRuler() != this) {
                                        int x = castles[i][k].getX();
                                        int y = castles[i][k].getY();
                                        double dist1 =obj.getDistanceTo(x,y);
                                        if (dist1 < dist) {
                                                dist = dist1;
                                                dir = obj.getDirectionTo(x,y);
                                        }
                                }
                        }
                }
                return dir;
        }

        int [][] dirs = new int[][]{
                //y, x
                new int []{0,0,MOVE_NONE},
                new int []{-1,0,MOVE_N},
                new int []{-1,1,MOVE_NE},
                new int []{0,1,MOVE_E},
                new int []{1,1,MOVE_SE},
                new int []{1,0,MOVE_S},
                new int []{1,-1,MOVE_SW},
                new int []{0,-1,MOVE_W},
                new int []{-1,-1,MOVE_NW},
        };
        int getMax(IObject p, int [][] map) {
                int x = p.getX();
                int y = p.getY();
                //test north
                int dir = MOVE_N;
                int max = -1;
                int [][] dirs = randomDir();
                for (int i = 0; i < dirs.length; i++) {
                        int nx = x+dirs[i][1];
                        int ny = y+dirs[i][0];
                        if (nx < 0 || ny <0 || ny >= World.HEIGHT || nx >= World.WID

                        }       else {
                                if (max < map[ny][nx]) {
                                        max = map[ny][nx];
                                        dir = dirs[i][2];
                                }
                        }
        }
        return max;
}
int getMaxDirection(IObject p, int [][] map) {
        int x = p.getX();
        int y = p.getY();
        //test north
        int dir = MOVE_NONE;
        int max = -1;
        int [][] dirs = randomDir();
        for (int i = 0; i < dirs.length; i++) {
                int nx = x+dirs[i][1];
                int ny = y+dirs[i][0];
                if (nx < 0 || ny <0 || ny >= World.HEIGHT || nx >= World.WID

                }       else {
                        if (max < map[ny][nx]) {
                                max = map[ny][nx];
                                dir = dirs[i][2];
                        }
                        }

                }
                return dir;
        }
        Random rand = new Random();
        int [][] randomDir() {
                int [][] rdir = new int[dirs.length][3];
                for (int i = 0 ; i < rdir.length ; i++) {
                        rdir[i] = dirs[i];
                }
                for (int i = 0 ; i < rdir.length ; i++) {
                        int j = rand.nextInt(rdir.length);
                        int [] tmp = rdir[i];
                        rdir[i] =rdir[j];
                        rdir[j] = tmp;
                }
                return rdir;
        }

}
```

References

[IBM04] IBM. *Code Ruler Manual*, 2004.

[Li04] Sing Li. Conquer medieval kingdoms with coderuler. 2004. http://www-106.ibm.com/developerworks/java/library/j-coderuler/.