|  |
| --- |
| package com.example.fred.mwesogame; |
|  |  |
|  | import android.media.MediaPlayer;//wabuluka davies |
|  | import android.app.Activity; |
|  | import android.graphics.drawable.AnimationDrawable; |
|  | import android.os.Bundle; |
|  | import android.view.View; |
|  | import android.widget.Button; |
|  | import android.widget.ImageButton; |
|  | import android.widget.ImageView; |
|  | import android.widget.TextView; |
|  |  |
|  |  |
|  | /\* animated button class \*/ |
|  | class AnimatedButton { |
|  |  |
|  | public ImageButton button; /\* which button we control \*/ |
|  | Game the\_game; /\* refernce to the game object \*/ |
|  | int cell\_number; /\* which cell do I refer to \*/ |
|  | boolean taken; /\* each cell can only be taken once \*/ |
|  | boolean allowed; |
|  |  |
|  | /\* X goes first, so we know which turn is which \*/ |
|  | static int global\_turn = 0; |
|  |  |
|  | int turn; |
|  |  |
|  | public void setImag(){ |
|  | if((turn%2)==0) |
|  | button.setImageResource(R.drawable.x7); |
|  | else |
|  | button.setImageResource(R.drawable.o7); |
|  | } |
|  |  |
|  |  |
|  | AnimatedButton(ImageButton b, Game g, int cell) { |
|  | /\* store references \*/ |
|  | button = b; |
|  | the\_game = g; |
|  | cell\_number = cell; |
|  | global\_turn = 0; |
|  | taken = false;//the button is still open i.e not occupied with either X or O |
|  | allowed = true; |
|  |  |
|  | if(global\_turn==0){ |
|  | switch(button.getId()){ |
|  | case R.id.ImageButton01: |
|  | taken = true; |
|  | allowed = false; |
|  | break; |
|  | case R.id.ImageButton02: |
|  | taken = true; |
|  | allowed = false; |
|  | break; |
|  | case R.id.ImageButton03: |
|  | taken = true; |
|  | allowed = false; |
|  | break; |
|  | case R.id.ImageButton07: |
|  | taken = true; |
|  | allowed = false; |
|  | break; |
|  | case R.id.ImageButton08: |
|  | taken = true; |
|  | allowed = false; |
|  | break; |
|  | case R.id.ImageButton09: |
|  | taken = true; |
|  | allowed = false; |
|  | break; |
|  | } |
|  | } |
|  |  |
|  | /\* event handler \*/ |
|  | button.setOnClickListener(new Button.OnClickListener( ) { |
|  | public void onClick(View v) { |
|  | /\* prevent double takes \*/ |
|  | if(taken || global\_turn<0 ) {//if true(the button is already clicked) |
|  | button.setImageResource(android.R.color.transparent); |
|  | button.setTag("empty"); |
|  | the\_game.status.setText("Choose another cell"); |
|  | // the\_game.status.setText("This Cell is Already Taken"); |
|  | taken = false; |
|  | return; |
|  | } |
|  | if(taken || global\_turn>0){ |
|  | button.setImageResource(android.R.color.transparent); |
|  | button.setTag("empty"); |
|  | the\_game.status.setText("Choose another cell"); |
|  | taken = false; |
|  | } |
|  | if(button.getTag()=="empty" || global\_turn>0){ |
|  | allowed = true; |
|  | if(allowed){ |
|  | turn = global\_turn;//turn is equal to zero which is the even number that corresponds to X |
|  | global\_turn++;//global turn is incremented to one which is odd number that corresponds to O |
|  | setImag(); |
|  | taken = true; |
|  | the\_game.update(cell\_number,turn); |
|  | }else{ |
|  | button.setImageResource(android.R.color.transparent); |
|  | button.setTag("empty"); |
|  | the\_game.status.setText("Choose another cell"); |
|  | allowed = false; |
|  | taken = false; |
|  | } |
|  | button.setTag(""); |
|  | } |
|  | if(!taken ){ |
|  | if(allowed) { |
|  | turn = global\_turn;//turn is equal to zero which is the even number that corresponds to X |
|  | global\_turn++;//global turn is incremented to one which is odd number that corresponds to O |
|  | taken = true; |
|  | setImag(); |
|  | the\_game.update(cell\_number,turn); |
|  | } |
|  | } |
|  | } |
|  | }); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | public class Game extends Activity { |
|  |  |
|  | /\* number of players \*/ |
|  | int num\_players; |
|  |  |
|  | enum Cell { |
|  | X, |
|  | O |
|  | } |
|  |  |
|  | enum Outcome { |
|  | NONE, |
|  | P1\_WON, |
|  | P2\_WON, |
|  | COMPUTER\_WON |
|  | } |
|  |  |
|  | private Cell [] cells; |
|  |  |
|  | public void setupGame( ) { |
|  | cells = new Cell [9]; |
|  | //now all cells are open meaning they dont contain anything |
|  | } |
|  |  |
|  | public boolean checkTriple(int c1, int c2, int c3, Cell value) { |
|  | //to test whether the cells c1, c2, c3 contain the same value |
|  | if((cells[c1] == cells[c2]) && (cells[c2] == cells[c3]) && (cells[c3] == value)) |
|  | return true; |
|  | else |
|  | return false; |
|  | } |
|  |  |
|  | public Outcome checkGame( ) { |
|  | /\* I wish Java had macros :( \*/ |
|  |  |
|  | /\* top row \*/ |
|  | if(checkTriple(0, 1, 2, Cell.X)) return Outcome.P1\_WON;//if the cell(0, 1 and 2) all have same value X |
|  | //then outcome is player one wins. |
|  | else if(checkTriple(0, 1, 2, Cell.O)) {//else player two or computer wins |
|  | return (num\_players == 1) ? Outcome.COMPUTER\_WON : Outcome.P2\_WON; |
|  | } |
|  |  |
|  | /\* middle row \*/ |
|  | if(checkTriple(3, 4, 5, Cell.X)) return Outcome.P1\_WON; |
|  | else if(checkTriple(3, 4, 5, Cell.O)) { |
|  | return (num\_players == 1) ? Outcome.COMPUTER\_WON : Outcome.P2\_WON; |
|  | } |
|  |  |
|  | /\* bottom row \*/ |
|  | if(checkTriple(6, 7, 8, Cell.X)) return Outcome.P1\_WON; |
|  | else if(checkTriple(6, 7, 8, Cell.O)) { |
|  | return (num\_players == 1) ? Outcome.COMPUTER\_WON : Outcome.P2\_WON; |
|  | } |
|  |  |
|  | /\* left col \*/ |
|  | if(checkTriple(0, 3, 6, Cell.X)) return Outcome.P1\_WON; |
|  | else if(checkTriple(0, 3, 6, Cell.O)) { |
|  | return (num\_players == 1) ? Outcome.COMPUTER\_WON : Outcome.P2\_WON; |
|  | } |
|  |  |
|  | /\* middle col \*/ |
|  | if(checkTriple(1, 4, 7, Cell.X)) return Outcome.P1\_WON; |
|  | else if(checkTriple(1, 4, 7, Cell.O)) { |
|  | return (num\_players == 1) ? Outcome.COMPUTER\_WON : Outcome.P2\_WON; |
|  | } |
|  |  |
|  | /\* right col \*/ |
|  | if(checkTriple(2, 5, 8, Cell.X)) return Outcome.P1\_WON; |
|  | else if(checkTriple(2, 5, 8, Cell.O)) { |
|  | return (num\_players == 1) ? Outcome.COMPUTER\_WON : Outcome.P2\_WON; |
|  | } |
|  |  |
|  | /\* / \*/ |
|  | if(checkTriple(2, 4, 6, Cell.X)) return Outcome.P1\_WON; |
|  | else if(checkTriple(2, 4, 6, Cell.O)) { |
|  | return (num\_players == 1) ? Outcome.COMPUTER\_WON : Outcome.P2\_WON; |
|  | } |
|  |  |
|  | /\* \ \*/ |
|  | if(checkTriple(0, 4, 8, Cell.X)) return Outcome.P1\_WON; |
|  | else if(checkTriple(0, 4, 8, Cell.O)) { |
|  | return (num\_players == 1) ? Outcome.COMPUTER\_WON : Outcome.P2\_WON; |
|  | } |
|  | return Outcome.NONE; |
|  | } |
|  |  |
|  | public void finishGame(Outcome out, int call\_no, String p1\_name, String p2\_name) { |
|  |  |
|  | /\* new status \*/ |
|  | final TextView status2 = (TextView) findViewById(R.id.TextView01); |
|  | final TextView gameover = (TextView) findViewById(R.id.gameover); |
|  |  |
|  | if(call\_no == 1) { |
|  | /\* set the win animation \*/ |
|  | setContentView(R.layout.win); |
|  | status2.setText(status.getText( )); |
|  |  |
|  | final ImageView img = (ImageView) findViewById(R.id.ImageView01); |
|  | //noinspection ResourceType |
|  | AnimationDrawable ad = (AnimationDrawable) getBaseContext( ).getResources( ) |
|  | .getDrawable(R.anim.fworks); |
|  |  |
|  | img.setImageDrawable(ad); |
|  | ad.start( ); |
|  |  |
|  |  |
|  | /\* this dialog will call back with call\_no = 2 and p1\_name in place\*/ |
|  | NameDialog diag = new NameDialog(this, out, 1, "", num\_players); |
|  | diag.show( ); |
|  |  |
|  | return; |
|  | } else if(call\_no == 2) { |
|  | if(num\_players == 2) { |
|  | /\* this dialog will call back with call\_no = 3 and p1\_name and p2\_name in place\*/ |
|  | NameDialog diag = new NameDialog(this, out, 2, p1\_name, num\_players); |
|  | diag.show( ); |
|  | return; |
|  | } |
|  | } |
|  | } |
|  |  |
|  |  |
|  |  |
|  | public boolean canWin(int index, Cell player) { |
|  | /\* save old val \*/ |
|  | Cell old = cells[index]; |
|  |  |
|  | /\* overwrite \*/ |
|  | cells[index] = player; |
|  |  |
|  | /\* does this result in win? \*/ |
|  | boolean can; |
|  | Outcome out = checkGame( ); |
|  | if((out == Outcome.NONE)) |
|  | can = false; |
|  | else |
|  | can = true; |
|  |  |
|  | /\* restore value \*/ |
|  | cells[index] = old; |
|  |  |
|  | return can; |
|  | } |
|  |  |
|  | public int rankMove(int index) { |
|  | /\* if the space is taken, it is a 0 \*/ |
|  |  |
|  | /\* check if we can win here \*/ |
|  | if(canWin(index, Cell.O)) |
|  | return 100; |
|  |  |
|  | /\* check if opponent could win here \*/ |
|  | if(canWin(index, Cell.X)) |
|  | return 50; |
|  |  |
|  | /\* center square \*/ |
|  | if(index == 4) |
|  | return 25; |
|  |  |
|  | /\* corner \*/ |
|  | if((index == 0) || (index == 2) || (index == 6) || (index == 8)) |
|  | return 10; |
|  |  |
|  | /\* meh \*/ |
|  | return 5; |
|  |  |
|  | } |
|  |  |
|  | public void doAi( ) { |
|  | int rankings [] = new int[9]; |
|  |  |
|  | /\* get the rankings \*/ |
|  | for(int i = 0; i < 9; i++) { |
|  | rankings[i] = rankMove(i); |
|  | } |
|  |  |
|  | /\* choose best ranking \*/ |
|  | int best\_ranking = 0; |
|  | for(int i = 0; i < 9; i++) { |
|  | if(rankings[i] > rankings[best\_ranking]) |
|  | best\_ranking = i; |
|  | } |
|  |  |
|  | /\* go with the best ranking \*/ |
|  | buttons[best\_ranking].button.performClick( ); |
|  | } |
|  |  |
|  | /\* called when a button is clicked \*/ |
|  | //cell = integer for cell number i.e ( 0 to 8 ) |
|  | //turn = integer for identifying player's turn |
|  | public void update(int cell, int turn) { |
|  | /\* set the cell \*/ |
|  | cells[cell] = ((turn % 2) == 0) ? Cell.X : Cell.O; |
|  | //If Cell of index i.e (0 to 8) is even return X or else return O |
|  |  |
|  | /\* check for a winner \*/ |
|  | Outcome o; |
|  | switch(o = checkGame( )) { |
|  | case P1\_WON: |
|  | status.setText("Player 1 Won!!"); |
|  | finishGame(o, 1, "", ""); |
|  | break; |
|  | case P2\_WON: |
|  | status.setText("Player 2 Won!!"); |
|  | finishGame(o, 1, "", ""); |
|  | break; |
|  | case COMPUTER\_WON: |
|  | status.setText("You Have Lost!!"); |
|  | finishGame(o, 1, "", ""); |
|  | break; |
|  | case NONE: |
|  | status.setText(((turn % 2) == 0) ? "O's Turn" : "X's Turn"); |
|  | break; |
|  | } |
|  |  |
|  | /\* time to do computer move if needed \*/ |
|  | if((num\_players == 1) && ((turn % 2) == 0) && (o == Outcome.NONE)) |
|  | doAi( ); |
|  | } |
|  |  |
|  |  |
|  | /\* our buttons \*/ |
|  | private AnimatedButton buttons [] = new AnimatedButton [9]; |
|  |  |
|  | /\* text view for debugging \*/ |
|  | public TextView status; |
|  |  |
|  | /\* setup buttons and interface \*/ |
|  | public void onCreate(Bundle savedInstanceState) { |
|  | super.onCreate(savedInstanceState); |
|  | setContentView(R.layout.gamelayout); |
|  |  |
|  | //to instatate the media element when the game is started.wabuluka Davies |
|  | MediaPlayer mdPlayer = MediaPlayer.create(this, R.raw.splash); |
|  | mdPlayer.start(); |
|  |  |
|  | setupGame( ); |
|  |  |
|  | status = (TextView) findViewById(R.id.TextView01); |
|  |  |
|  | /\* set number of players \*/ |
|  | if(getIntent( ).getType( ).equalsIgnoreCase("1")) |
|  | num\_players = 1; |
|  | else |
|  | num\_players = 2; |
|  |  |
|  | status.setText("X's Turn"); |
|  |  |
|  |  |
|  | /\* setup buttons \*/ |
|  | int button\_ids [] = {R.id.ImageButton01, R.id.ImageButton02, R.id.ImageButton03, |
|  | R.id.ImageButton04, R.id.ImageButton05, R.id.ImageButton06, |
|  | R.id.ImageButton07, R.id.ImageButton08, R.id.ImageButton09}; |
|  |  |
|  | for(int i=0; i<9; i++) { |
|  | buttons[i] = new AnimatedButton((ImageButton) findViewById(button\_ids[i]), this, i); |
|  | } |
|  | }  //The method below enables a user to shift tiles from one position to another free position  public void setBefore(){  String string = the\_game.status.getText().toString();  if((string.equalsIgnoreCase("X's Turn") && cells[cell\_number]==Cell.X) || (string.equalsIgnoreCase("O's Turn") && cells[cell\_number]==Cell.O)){  button.setImageResource(android.R.color.transparent);  the\_game.status.setText("Choose another cell");  cells[cell\_number]=Cell.E;  save = true;  dont=1;  }else{  dont=0;  }  }  public void setAfter(){  if(dont==1){  turn = global\_turn;//turn is equal to zero which is the even number that corresponds to X(green)  global\_turn++;//global turn is incremented to one which is odd number that corresponds to O(yellow)  setImag();  save = false;  if((turn % 2) == 0) cells[cell\_number] = Cell.X; else cells[cell\_number] = Cell.O;  the\_game.update(cell\_number, turn);  }  } |
|  | } |