Logotipo

Descripción generada automáticamenteUn dibujo de una persona

Descripción generada automáticamente con confianza bajaInstituto Tecnológico Superior de Calkiní en el Estado de Campeche

SCD-1027 Tópicos avanzados de programación

|| PRÁCTICA 6 Desarrollo de Pong para dispositivos móviles ||

**Patrón de fondo

Descripción generada automáticamente**Ingeniería en sistemas computacionales

**Ing. José Luis**

**Lira Turriza**

**Diego Manuel Chi Tzab |7262|**

**4° Semestre Grupo A**

# PRÁCTICA No. 6.- Desarrollo de Pong para dispositivos móviles

#### INTRODUCCIÓN

En esta práctica se describe como desarrollar y publicar un juego básico Pong que corre bajo Android. Se desarrolla el juego con modo de un único jugador. Pong es el hola mundo del desarrollo de los juegos y fue originalmente desarrollado como una versión de ping pong electrónico. La idea es mantener la bola en juego y esperar que el contrincante no lo haga.

En esta práctica permite ejemplificar Actividades, Vistas y Canvas para desarrollar animaciones simples para Android.

#### -OBJETIVO

El estudiante que termine la práctica podrá implementar animaciones a través de clases de Canvas sobre Actividades y vistas implementadas en Android.

#### LUGAR

AULA

#### -SEMANA DE EJECUCIÓN

Semana CATORCE (Parcial 3)

#### MATERIAL Y EQUIPO

* + Sistema Operativo
  + Procesador de Textos
  + Software para el desarrollo de aplicaciones “eclipse”.
  + Cañón
  + Plumones
  + Pizarrón.
  + Equipos de cómputo para todos los estudiantes de la asignatura.

#### DESARROLLO DE LA PRÁCTICA

Desarrollar el juego del PingPong. La mecánica del juego es bien sencilla. Nosotros manejamos una especie de raqueta que debe golpear una pelota para marcar un tanto en el lado contrario del terreno de juego. Como contrincante tenemos a otra raqueta controlada por la máquina y que intentará hacer lo mismo que nosotros para marcarse ella el tanto. El primero de los dos que llegue a 10 tantos gana el partido.

1. Crea una clase PingPong que herede de MIDlet con los siguientes atributos:

private Display pantalla; private Menu m;

private Tablero t; private Resultado r; private Dificultad d; private int nivel = 1;

private NuevoRecord nr;

1. Para cada atributo declarado crea un método set y uno get que controle los valores de cada

uno.

1. Crea la clase Menu que herede de List e implemente de CommandListener con dos atributos: midlet y salir, de tipo PingPong y Command respectivamente con un constructor como se muestra a continuación:

public Menu(PingPong m){ super("Menu",List.IMPLICIT); this.append("Jugar",null); this.append("Dificultad",null); this.append("Resultados",null); this.midlet = m;

salir = new Command("Salir",Command.EXIT,1); this.addCommand(salir); this.setCommandListener(this);

}

1. De igual manera crea la clase Dificultad con las mismas herencias que Menu con el siguiente

constructor:

public Dificultad(PingPong m){ super("Dificultad",List.EXCLUSIVE); this.append("Baja",null); this.append("Normal",null); this.append("Alta",null); this.setSelectedIndex(0,true);

midlet = m;

atras = new Command("Atras",Command.BACK,1); aceptar = new Command("Aceptar",Command.OK,1);

this.addCommand(aceptar); this.addCommand(atras); this.setCommandListener(this);

1. Crea la clase NuevoRecord que herede de Form con CommandListener con los siguientes atributos y constructor:

private TextField txtNombre; private Command aceptar; private PingPong midlet; private long puntuacion;

public NuevoRecord(PingPong m){ super("Nuevo Record"); midlet = m;

puntuacion = 0;

txtNombre = new TextField("Nombre","",3,TextField.ANY); aceptar = new Command("Aceptar",Command.OK,1); this.append(txtNombre);

this.addCommand(aceptar); this.setCommandListener(this);

}

1. Crea la clase Pelota y la clase Paleta que herende de Sprite que pinte la pelota del juego y la

raqueta con las imágenes de los Anexos, la pelota debe ser dibujada hacia una dirección hacia x o y.

1. Por último crea una clase Tablero que herede de GameCanvas e implemente Runnable y CommandListener, esta clase será la principal y que reúne las clases anteriores. Diséñalas de acuerdo al bosquejo de métodos siguientes:

crearFondo(): Nos crea el fondo de pantalla. crearPelota(): Nos crea el objeto pelota. crearPaleta(): Nos crea el objeto paleta. inicializar(): Nos inicializa los elementos del juego. start(): Comenzamos el juego.

run(): Posee el cuerpo principal del juego.

verColisiones(): Detecta cualquier colisión que produzca la pelota y actúa en consecuencia. calcularTrayectoria(Paleta pal): Calcula la nueva trayectoria de la pelota tras ocurrir una colisión. verEntrada(): Aquí se detecta cualquier pulsación de teclas que efectúe el usuario. moverPelota(): Movemos la pelota según la dirección que posea.

dibujar(Graphics g): Dibujamos todos los elementos por pantalla. mostrarResultados(int r1, int r2): Nos muestra el resultado por pantalla tras un tanto.

1. Muestra las pantallas diseñadas en el emulador de J2ME

#### EVALUACIÓN Y RESULTADOS

Se describe la forma de evaluar la práctica desarrollada mediante la entrega del Informe Técnico solicitado por el Profesor, el cual puede contener tablas, planos, prototipos, gráficas, diagramas o dibujos, observaciones, conclusiones, cuestionario y referencias.

Para la siguiente practica decidimos usar la aplicación Android Studio ya que está herramienta nos ofrece varias herramientas aparte de una fácil manipulación de las mismas

1. Principalmente crearemos el Main donde ejecutaremos todo al igual que será la conexión entre todas las clases que vayamos creando

|  |
| --- |
| package com.example.practica6\_pingpong;   import android.app.Activity; import android.os.Bundle; import android.view.Menu; import android.view.MenuItem; import android.view.Window; import android.widget.TextView;  public class MainActivity extends Activity {  private GameSurfaceView pongSurfaceView;   */\*\*  \* Called when the activity is first created.jhbhjghjgjgjh  \*/* @Override  public void onCreate(Bundle savedInstanceState)  {  super.onCreate(savedInstanceState);  requestWindowFeature(Window.FEATURE\_NO\_TITLE);  setContentView(R.layout.activity\_main);  pongSurfaceView = (GameSurfaceView) findViewById(R.id.game);  pongSurfaceView.setTextView((TextView) findViewById(R.id.text));  }   private static final int MENU\_PAUSE = 4;  private static final int MENU\_RESUME = 5;  private static final int MENU\_START\_1P = 6;  private static final int MENU\_START\_2P = 7;  private static final int MENU\_START\_0P = 8;  private static final int MENU\_SOUND\_ON = 11;   @Override  public boolean onCreateOptionsMenu(Menu menu)  {  super.onCreateOptionsMenu(menu);  menu.add(0, MENU\_START\_1P, 0, R.string.menu\_start\_1p);  menu.add(0, MENU\_START\_2P, 0, R.string.menu\_start\_2p);  menu.add(0, MENU\_START\_0P, 0, R.string.menu\_start\_0p);  menu.add(0, MENU\_PAUSE, 0, R.string.menu\_pause);  menu.add(0, MENU\_RESUME, 0, R.string.menu\_resume);  menu.add(0, MENU\_SOUND\_ON, 0, R.string.menu\_sound);  return true;  }   @Override  public boolean onMenuOpened(int featureId, Menu menu)  {  GameThread androidPongThread = pongSurfaceView.getGamePongThread();  super.onMenuOpened(featureId, menu);  androidPongThread.pause();  return true;  }   @Override  public boolean onOptionsItemSelected(MenuItem item)  {  GameThread androidPongThread = pongSurfaceView.getGamePongThread();  switch (item.getItemId())  {  case MENU\_START\_1P:  androidPongThread.doStart1p();  return true;  case MENU\_START\_2P:  androidPongThread.doStart2p();  return true;  case MENU\_START\_0P:  androidPongThread.doStart0p();  return true;  case MENU\_PAUSE:  androidPongThread.pause();  return true;  case MENU\_RESUME:  androidPongThread.unpause();  return true;  case MENU\_SOUND\_ON:  androidPongThread.toggleSound();  return true;  }   return false;  }   protected void onPause()  {  super.onPause();  GameThread androidPongThread = pongSurfaceView.getGamePongThread();  androidPongThread.pause(); // pause game when Activity pauses  }   @Override  protected void onResume()  {  super.onResume();  GameThread androidPongThread = pongSurfaceView.getGamePongThread();  androidPongThread.resumeGame();  }   protected void onDestroy()  {  super.onDestroy();  SoundManager.cleanup();  } } |

1. Posteriormente haremos la parte Visual del juego

|  |
| --- |
| package com.example.practica6\_pingpong;  import android.content.Context; import android.os.Bundle; import android.os.Handler; import android.os.Message; import android.util.AttributeSet; import android.view.MotionEvent; import android.view.SurfaceHolder; import android.view.SurfaceView; import android.widget.TextView; import android.widget.Toast;  public class GameSurfaceView extends SurfaceView implements SurfaceHolder.Callback {  private GameThread gameThread;  private TextView statusText;  private SurfaceHolder holder;  private Context context;   public GameSurfaceView(Context context, AttributeSet attrs)  {  super(context, attrs);  this.context = context;  this.holder = getHolder();  holder.addCallback(this);  setFocusable(true);  gameThread = CreateNewGameThread();  }   private GameThread CreateNewGameThread()  {  Toast toast = Toast.makeText(context, "Select Menu for a new game", Toast.LENGTH\_LONG);  toast.show();   return new GameThread(holder, context, new Handler()  {  @Override  public void handleMessage(Message m)  {  Bundle bundle = m.getData();  if (bundle.containsKey("viz"))  {  statusText.setVisibility(m.getData().getInt("viz"));  statusText.setText(m.getData().getString("text"));  }  else if (bundle.containsKey("toast"))  {  Toast toast = Toast.makeText(context, bundle.getString("toast"), Toast.LENGTH\_LONG);  toast.show();  }  }  });  }    public void setTextView(TextView textView)  {  statusText = textView;  }   public void surfaceCreated(SurfaceHolder surfaceHolder)  {  gameThread.setRunning(true);  try  {  gameThread.start();  }  catch (Exception error)  {  gameThread = CreateNewGameThread();  gameThread.start();  gameThread.setRunning(true);  }  }   public void surfaceChanged(SurfaceHolder surfaceHolder, int format, int width, int height)  {  gameThread.setSurfaceSize(width, height);  }   public void surfaceDestroyed(SurfaceHolder surfaceHolder)  {  gameThread.setRunning(false);  boolean retry = true;  while (retry)  {  try  {  gameThread.join();  retry = false;  }  catch (InterruptedException e)  {  }  }  }   public GameThread getGamePongThread()  {  return gameThread;  }   @Override  public void onWindowFocusChanged(boolean hasWindowFocus)  {  if (!hasWindowFocus)  {  gameThread.pause();  }  }   public boolean onTouchEvent(MotionEvent event)  {  float xPosition1 = 0;  float yPosition1 = 0;  float xPosition2 = 0;  float yPosition2 = 0;   for (int pointerIndex = 0; pointerIndex < event.getPointerCount(); pointerIndex++)  {  if (pointerIndex == 0)  {  xPosition1 = event.getX(pointerIndex);  yPosition1 = event.getY(pointerIndex);  }   if (pointerIndex == 1)  {  xPosition2 = event.getX(pointerIndex);  yPosition2 = event.getX(pointerIndex);  }  }   switch (event.getAction())  {  case MotionEvent.ACTION\_MOVE:  gameThread.setBattPosition(xPosition1, yPosition1, xPosition2, yPosition2);  break;  }  return true;  } }  package com.example.practica6\_pingpong;  import android.content.Context; import android.os.Bundle; import android.os.Handler; import android.os.Message; import android.util.AttributeSet; import android.view.MotionEvent; import android.view.SurfaceHolder; import android.view.SurfaceView; import android.widget.TextView; import android.widget.Toast;  public class GameSurfaceView extends SurfaceView implements SurfaceHolder.Callback {  private GameThread gameThread;  private TextView statusText;  private SurfaceHolder holder;  private Context context;   public GameSurfaceView(Context context, AttributeSet attrs)  {  super(context, attrs);  this.context = context;  this.holder = getHolder();  holder.addCallback(this);  setFocusable(true);  gameThread = CreateNewGameThread();  }   private GameThread CreateNewGameThread()  {  Toast toast = Toast.makeText(context, "Select Menu for a new game", Toast.LENGTH\_LONG);  toast.show();   return new GameThread(holder, context, new Handler()  {  @Override  public void handleMessage(Message m)  {  Bundle bundle = m.getData();  if (bundle.containsKey("viz"))  {  statusText.setVisibility(m.getData().getInt("viz"));  statusText.setText(m.getData().getString("text"));  }  else if (bundle.containsKey("toast"))  {  Toast toast = Toast.makeText(context, bundle.getString("toast"), Toast.LENGTH\_LONG);  toast.show();  }  }  });  }    public void setTextView(TextView textView)  {  statusText = textView;  }   public void surfaceCreated(SurfaceHolder surfaceHolder)  {  gameThread.setRunning(true);  try  {  gameThread.start();  }  catch (Exception error)  {  gameThread = CreateNewGameThread();  gameThread.start();  gameThread.setRunning(true);  }  }   public void surfaceChanged(SurfaceHolder surfaceHolder, int format, int width, int height)  {  gameThread.setSurfaceSize(width, height);  }   public void surfaceDestroyed(SurfaceHolder surfaceHolder)  {  gameThread.setRunning(false);  boolean retry = true;  while (retry)  {  try  {  gameThread.join();  retry = false;  }  catch (InterruptedException e)  {  }  }  }   public GameThread getGamePongThread()  {  return gameThread;  }   public void onWindowFocusChanged(boolean hasWindowFocus)  {  if (!hasWindowFocus)  {  gameThread.pause();  }  }   public boolean onTouchEvent(MotionEvent event)  {  float xPosition1 = 0;  float yPosition1 = 0;  float xPosition2 = 0;  float yPosition2 = 0;   for (int pointerIndex = 0; pointerIndex < event.getPointerCount(); pointerIndex++)  {  if (pointerIndex == 0)  {  xPosition1 = event.getX(pointerIndex);  yPosition1 = event.getY(pointerIndex);  }   if (pointerIndex == 1)  {  xPosition2 = event.getX(pointerIndex);  yPosition2 = event.getX(pointerIndex);  }  }   switch (event.getAction())  {  case MotionEvent.ACTION\_MOVE:  gameThread.setBattPosition(xPosition1, yPosition1, xPosition2, yPosition2);  break;  }  return true;  } } |

1. Posterior agregaremos la clase DrawableResourceCollection

|  |
| --- |
| package com.example.practica6\_pingpong;  import android.graphics.drawable.Drawable;  import java.util.LinkedList;  public class DrawableResourceCollection extends LinkedList<Drawable>{  } |

1. Empezaremos a agregar la pelota, ya que un juego de ping pong sin la pelota no sería un juego

|  |
| --- |
| package com.example.practica6\_pingpong;  public class Ball extends Sprite {  private VelocityGenerator velocityGenerator;   public Ball(DrawableResourceCollection drawableResourceCollection, int canvasWidth, int canvasHeight, VelocityGenerator velocityGenerator)  {  super(drawableResourceCollection, canvasWidth, canvasHeight, 12, true);  this.drawableResourceCollection = drawableResourceCollection;  this.velocityGenerator = velocityGenerator;  }   public void reverseXVelocity()  {  this.velocity.ReverseX();  this.reverseAnimation();  }   public void setInitialVelocity()  {  this.velocity = velocityGenerator.GenerateInitialVelocity();  }   public void generateNewVelocityDown()  {  this.velocity = this.velocityGenerator.GenerateNewReverseDown(velocity);  }   public void generateNewVelocityUp()  {  this.velocity = this.velocityGenerator.GenerateNewReverseUp(velocity);  }  } |

1. Posterior pondremos los Sprite y sus funcionalidades en el juego

|  |
| --- |
| package com.example.practica6\_pingpong;  import android.graphics.Bitmap; import android.graphics.Canvas; import android.graphics.drawable.BitmapDrawable; import android.os.SystemClock;  public class Sprite {  protected double xPosition;  protected double yPosition;  protected DrawableResourceCollection drawableResourceCollection;  private Bitmap internalBitmap;  private int canvasWidth;  private int canvasHeight;   protected Velocity velocity;  private int currentFrame;  private double animationSpeedInFpms;  private long startTimeInMillis;  private boolean animationForward;   public Sprite(DrawableResourceCollection drawableResourceCollection, int canvasWidth, int canvasHeight, int animationSpeedInFps, boolean animationForward)  {  this.canvasWidth = canvasWidth;  this.canvasHeight = canvasHeight;  this.drawableResourceCollection = drawableResourceCollection;  this.animationSpeedInFpms = animationSpeedInFps / 1000.0;  this.internalBitmap = ((BitmapDrawable) this.drawableResourceCollection.get(0).getCurrent()).getBitmap();  this.startTimeInMillis = SystemClock.*uptimeMillis*();  this.animationForward = animationForward;  }   public double getxPosition()  {  return xPosition;  }   public void draw(Canvas canvas)  {  drawableResourceCollection.get(currentFrame).setBounds((int) xPosition, (int) yPosition, (int) xPosition + getWidth(), (int) yPosition + getHeight());  drawableResourceCollection.get(currentFrame).draw(canvas);  currentFrame = GetNewFrame();  }   private int GetNewFrame()  {  int drawableResourceCollectionSize = drawableResourceCollection.size();  long elapsedInMillis = SystemClock.*uptimeMillis*() - startTimeInMillis;  int frame = (int) ((elapsedInMillis \* animationSpeedInFpms) % drawableResourceCollectionSize);  if (animationForward)  return frame;  else  return drawableResourceCollectionSize - (frame + 1);  }   public void canvasChanges(int canvasWidth, int canvasHeight)  {  this.canvasWidth = canvasWidth;  this.canvasHeight = canvasHeight;  }   public void setxPosition(int xPosition)  {  this.xPosition = xPosition;  }   public double getyPosition()  {  return yPosition;  }   public void setyPosition(int yPosition)  {  this.yPosition = yPosition;  }   public void reverseAnimation()  {  animationForward = !animationForward;  }   public int getWidth()  {  return drawableResourceCollection.get(currentFrame).getIntrinsicWidth();  }   public int getHeight()  {  return drawableResourceCollection.get(currentFrame).getIntrinsicHeight();  }   public void Move(double frameTime)  {  double newXPosition = velocity.getNewXPosition(xPosition, frameTime);  if (newXPosition + getWidth() < (canvasWidth - 4) && newXPosition > 4)  xPosition = newXPosition;   yPosition = velocity.getNewYPosition(yPosition, frameTime);  }   public boolean CollidesWith(Sprite sprite, double frameTime)  {  int left1, left2;  int right1, right2;  int top1, top2;  int bottom1, bottom2;   left1 = (int) this.getNewXPosition(frameTime);  left2 = (int) sprite.getNewXPosition(frameTime);  right1 = left1 + this.getWidth();  right2 = left2 + sprite.getWidth();  top1 = (int) this.getNewYPosition(frameTime);  top2 = (int) sprite.getNewYPosition(frameTime);  bottom1 = top1 + this.getHeight();  bottom2 = top2 + sprite.getHeight();   if (bottom1 < top2)  return false;  if (top1 > bottom2)  return false;  if (right1 < left2)  return false;  if (left1 > right2)  return false;   int over\_bottom;  int over\_top;  int over\_right;  int over\_left;   if (bottom1 > bottom2)  over\_bottom = bottom2;  else  over\_bottom = bottom1;   if (top1 < top2)  over\_top = top2;  else  over\_top = top1;   if (right1 > right2)  over\_right = right2;  else  over\_right = right1;   if (left1 < left2)  over\_left = left2;  else  over\_left = left1;   for (int xPos = over\_left; xPos < over\_right; xPos++)  {  for (int yPos = over\_top; yPos < over\_bottom; yPos++)  {  int thisColor = this.getPixel(xPos, yPos);  int spriteColor = sprite.getPixel(xPos, yPos);  if (thisColor != 0 && spriteColor != 0)  {  return true;  }  }  }   return false;  }   public double getNewXPosition(double frameTime)  {  return velocity.getNewXPosition(xPosition, frameTime);  }   public double getNewYPosition(double frameTime)  {  return velocity.getNewYPosition(yPosition, frameTime);  }   private int getPixel(int x, int y)  {  int xPosInBitmap;  if (x > getxPosition())  xPosInBitmap = x - (int) getxPosition();  else  xPosInBitmap = (int) getxPosition() - x;   int yPosInBitmap;  if (y > getyPosition())  yPosInBitmap = y - (int) getyPosition();  else  yPosInBitmap = (int) getyPosition() - y;   if (x >= internalBitmap.getWidth() || y >= internalBitmap.getHeight())  return 1;   if (yPosInBitmap < this.internalBitmap.getHeight() && xPosInBitmap < internalBitmap.getWidth())  return this.internalBitmap.getPixel(xPosInBitmap, yPosInBitmap);  else  return 0;  }   public void center()  {  centerHorizontal();  centerVertical();  }   public void centerHorizontal()  {  setxPosition(canvasWidth / 2 - getWidth() / 2);  }   public void centerVertical()  {  setyPosition(canvasHeight / 2 - getHeight() / 2);  }   public boolean IsMaximumRight(int canvasWidth)  {  return getxPosition() >= (canvasWidth - getWidth());  }   public void SetMaximumRight(int canvasWidth)  {  setxPosition(canvasWidth - getWidth());  }   public void setVelocity(Velocity velocity)  {  this.velocity = velocity;  }   public void setPreviousLocation(double frameTime)  {  xPosition = velocity.getPreviousXPosition(xPosition, frameTime);  yPosition = velocity.getPreviousYPosition(yPosition, frameTime);  }   public boolean IsOutOfXBounds(double frameTime)  {  return (getNewXPosition(frameTime) <= 7) || (getNewXPosition(frameTime) + getWidth() >= canvasWidth - 7);  }   public boolean IsOutOfLowerBounds(double frameTime)  {  return ((int) getNewYPosition(frameTime) + getHeight() >= canvasHeight - 10);  }   public boolean IsOutOfUpperBounds(double frameTime)  {  return getNewYPosition(frameTime) <= 10;  } } |

1. Igual agregamos un puntaje para saber quién va ganando

|  |
| --- |
| package com.example.practica6\_pingpong;  public class Score {  private int player1Score;  private int player2Score;  private int scoreToWin;   public Score(int scoreToWin)  {  this.scoreToWin = scoreToWin;  }   public void Reset()  {  player1Score = 9;  player2Score = 0;  }   public void Player1Scored()  {  player1Score++;  }   public void Player2Scored()  {  player2Score++;  }   public String CreateScoreBoard()  {  return "Score " + player1Score + " : " + player2Score;  }   public String CreateWinnerBoard()  {  if (player1Score >= player2Score )  return "Player 1 has won the game";  else  return "Player 2 has won the game";  }   public boolean isGameFinished()  {  if (player1Score >= scoreToWin || player2Score >= scoreToWin)  return true;  else  return false;  } } |

1. Agregamos la funcionalidad de la velocidad del juego

|  |
| --- |
| package com.example.practica6\_pingpong;  public class Velocity {  double xVelocity;  double yVelocity;   public Velocity(double xVelocity, double yVelocity)  {  this.xVelocity = xVelocity;  this.yVelocity = yVelocity;  }   public void ReverseX()  {  this.xVelocity = -this.xVelocity;  }   public double getNewXPosition(double xPosition, double frameTime)  {  return xPosition + frameTime \* xVelocity;  }   public double getNewYPosition(double yPosition, double frameTime)  {  return yPosition + frameTime \* yVelocity;  }   public double getPreviousXPosition(double xPosition, double frameTime)  {  return xPosition - frameTime \* xVelocity;  }   public double getPreviousYPosition(double yPosition, double frameTime)  {  return yPosition - frameTime \* yVelocity;  } } |

1. En este agregamos igual el generador que le dará vida a esta parte

|  |
| --- |
| package com.example.practica6\_pingpong;  import java.util.Random;  public class VelocityGenerator {  private static final int *MaxSpeed* = 298;  private static final int *MinimumYSpeed* = 123;  private static final int *MinimumXSpeed* = 123;  private Random speedRandomizer;   public VelocityGenerator()  {  speedRandomizer = new Random(12313975);  }   public Velocity GenerateInitialVelocity()  {  return new Velocity(162.0, -152.0);  }   public Velocity GenerateNewReverseDown(Velocity velocity)  {  int xVelocity = GenerateNewXSpeed();  int yVelocity = GenerateNewYSpeed();   if (velocity.xVelocity > 0)  xVelocity = -xVelocity;   return new Velocity(xVelocity, yVelocity);  }   public Velocity GenerateNewReverseUp(Velocity velocity)  {  int xVelocity = GenerateNewXSpeed();  int yVelocity = GenerateNewYSpeed();   if (velocity.xVelocity > 0)  xVelocity = -xVelocity;   return new Velocity(xVelocity, -yVelocity);  }   private int GenerateNewYSpeed()  {  return speedRandomizer.nextInt(*MaxSpeed*) + *MinimumYSpeed*;  }   private int GenerateNewXSpeed()  {  return speedRandomizer.nextInt(*MaxSpeed*) + *MinimumXSpeed*;  } } |

1. Antes de cerrar agregamos unos sonidos para que el juego se vea coqueto

|  |
| --- |
| package com.example.practica6\_pingpong;  import android.content.Context; import android.media.AudioManager; import android.media.SoundPool;  import java.util.HashMap;  public class SoundManager {  static private SoundManager *\_instance*;  private static SoundPool *mSoundPool*;  private static HashMap<Integer, Integer> *mSoundPoolMap*;  private static AudioManager *mAudioManager*;  private static Context *mContext*;  private static boolean *playSound*;   private SoundManager()  {  *playSound* = true;  }   static synchronized public SoundManager getInstance()  {  if (*\_instance* == null)  *\_instance* = new SoundManager();  return *\_instance*;  }   public static void initSounds(Context theContext)  {  *mContext* = theContext;  *mSoundPool* = new SoundPool(4, AudioManager.*STREAM\_MUSIC*, 0);  *mSoundPoolMap* = new HashMap<Integer, Integer>();  *mAudioManager* = (AudioManager) *mContext*.getSystemService(Context.*AUDIO\_SERVICE*);  }   public static void loadSounds()  {  *mSoundPoolMap*.put(1, *mSoundPool*.load(*mContext*, R.raw.*hit*, 1));  *mSoundPoolMap*.put(2, *mSoundPool*.load(*mContext*, R.raw.*terminator*, 1));  *mSoundPoolMap*.put(3, *mSoundPool*.load(*mContext*, R.raw.*clap*, 1));  }   public static void playSound(int index, float speed)  {  if (*playSound*)  {  float streamVolume = *mAudioManager*.getStreamVolume(AudioManager.*STREAM\_MUSIC*);  streamVolume = streamVolume / *mAudioManager*.getStreamMaxVolume(AudioManager.*STREAM\_MUSIC*);  *mSoundPool*.play(*mSoundPoolMap*.get(index), streamVolume, streamVolume, 1, 0, speed);  }  }   public static void togglePlaySound()  {  *playSound* = !*playSound*;  if (!*playSound*)  {  for (int soundIndex : *mSoundPoolMap*.values())  {  *stopSound*(soundIndex);  }  }  }   public static void stopSound(int index)  {  *mSoundPool*.stop(*mSoundPoolMap*.get(index));  }   public static void cleanup()  {  *mSoundPool*.release();  *mSoundPool* = null;  *mSoundPoolMap*.clear();  *mAudioManager*.unloadSoundEffects();  *\_instance* = null;  } } |

1. Por último agregamos las funcionalidades del juego

|  |
| --- |
| package com.example.practica6\_pingpong;  import android.content.Context; import android.content.res.Resources; import android.graphics.Bitmap; import android.graphics.BitmapFactory; import android.graphics.Canvas; import android.os.Bundle; import android.os.Handler; import android.os.Message; import android.os.SystemClock; import android.view.SurfaceHolder; import android.view.View;  public class GameThread extends Thread {  private final SurfaceHolder surfaceHolder;  private Handler handler;  private boolean isRunning;   private Bitmap backgroundImage;  private Ball ball;  private Sprite battBottom;  private Sprite battTop;   private int canvasWidth;  private int canvasHeight;   private double delayTime = 0;  public int currentState;   public static final int *STATE\_PAUSE* = 2;  public static final int *STATE\_RUNNING\_1P* = 4;  public static final int *STATE\_RUNNING\_2P* = 5;  public static final int *STATE\_RUNNING\_0P* = 6;  private Score score;  private boolean shouldDiagnosticInformation = false;  private double frameTime;  private int previousState;   public GameThread(SurfaceHolder surfaceHolder, Context context, Handler handler)  {  this.surfaceHolder = surfaceHolder;  this.handler = handler;   Resources resources = context.getResources();  DrawableResourceCollection drawableCollection = CreateBallCollection(resources);   ball = new Ball(drawableCollection, canvasWidth, canvasHeight, new VelocityGenerator());  ball.setInitialVelocity();  ball.center();   DrawableResourceCollection battCollection = new DrawableResourceCollection();  battCollection.add(resources.getDrawable(R.drawable.*bat*));   battBottom = new Sprite(battCollection, canvasWidth, canvasHeight, 0, false);  battTop = new Sprite(battCollection, canvasWidth, canvasHeight, 0, false);  backgroundImage = BitmapFactory.*decodeResource*(resources, R.drawable.*background2*);  SetInitialBattPosition();   score = new Score(10);   SoundManager.*getInstance*();  SoundManager.*initSounds*(context);  SoundManager.*loadSounds*();  }   public GameThread(SurfaceHolder surfaceHolder){  this.surfaceHolder = surfaceHolder;  }   private DrawableResourceCollection CreateBallCollection(Resources resources)  {  DrawableResourceCollection drawableCollection = new DrawableResourceCollection();  drawableCollection.add(resources.getDrawable(R.drawable.*ball1*));  drawableCollection.add(resources.getDrawable(R.drawable.*ball2*));  drawableCollection.add(resources.getDrawable(R.drawable.*ball3*));  drawableCollection.add(resources.getDrawable(R.drawable.*ball4*));  drawableCollection.add(resources.getDrawable(R.drawable.*ball5*));  drawableCollection.add(resources.getDrawable(R.drawable.*ball6*));  drawableCollection.add(resources.getDrawable(R.drawable.*ball7*));  drawableCollection.add(resources.getDrawable(R.drawable.*ball8*));  drawableCollection.add(resources.getDrawable(R.drawable.*ball9*));  drawableCollection.add(resources.getDrawable(R.drawable.*ball10*));  drawableCollection.add(resources.getDrawable(R.drawable.*ball11*));  drawableCollection.add(resources.getDrawable(R.drawable.*ball12*));  return drawableCollection;  }   private void SetInitialBattPosition()  {  battBottom.centerHorizontal();  battBottom.setyPosition(canvasHeight - (canvasHeight / 100 \* 20));  battTop.centerHorizontal();  battTop.setyPosition((canvasHeight / 100) \* 5);  }   @Override  public void run()  {  long startTime = SystemClock.*uptimeMillis*();  while (isRunning)  {  Canvas canvas = null;  try  {  long currentTime = SystemClock.*uptimeMillis*();  frameTime = (currentTime - startTime) / 1000.0;  startTime = currentTime;   canvas = surfaceHolder.lockCanvas(null);  synchronized (surfaceHolder)  {  if ((currentState == *STATE\_RUNNING\_1P* || currentState == *STATE\_RUNNING\_0P* ||  currentState == *STATE\_RUNNING\_2P*) && delayTime <= 0)  {  CheckCollision();  CheckBallBounds();  AdjustBatts();  AdvanceBall(frameTime);  }   if (delayTime >= 0)  delayTime = delayTime - frameTime;   DrawScoreBoard();  doDraw(canvas);  }  }  finally  {  if (canvas != null)  {  surfaceHolder.unlockCanvasAndPost(canvas);  }  }  }  }   private void DrawScoreBoard()  {  Message msg = handler.obtainMessage();  Bundle b = new Bundle();  String scoreBoard = score.CreateScoreBoard();  if (shouldDiagnosticInformation)  scoreBoard += " FPS: " + (int) (1 / frameTime);  b.putString("text", scoreBoard);  b.putInt("viz", View.*VISIBLE*);  msg.setData(b);  handler.sendMessage(msg);  }   private void AddToastToQueue(String messageToShow)  {  Message msg = handler.obtainMessage();  Bundle b = new Bundle();  b.putString("toast", messageToShow);  msg.setData(b);  handler.sendMessage(msg);  }   private void AdjustBatts()  {  if (currentState == *STATE\_RUNNING\_0P* || currentState == *STATE\_RUNNING\_1P*)  {  AdjustBattPosition(battTop);  }   if (currentState == *STATE\_RUNNING\_0P*)  {  AdjustBattPosition(battBottom);  }  }   private void AdjustBattPosition(Sprite batt)  {  if (batt.getxPosition() < ball.getxPosition())  batt.setVelocity(new Velocity(100, 0));  if (batt.getxPosition() >= ball.getxPosition())  batt.setVelocity(new Velocity(-130, 0));  }   private void doDraw(Canvas canvas)  {  canvas.drawBitmap(backgroundImage, 0, 0, null);  ball.draw(canvas);  battBottom.draw(canvas);  battTop.draw(canvas);  }   private void CheckBallBounds()  {  if (ball.IsOutOfXBounds(frameTime))  {  ball.reverseXVelocity();  SoundManager.*playSound*(1, 2);  }   if (ball.IsOutOfLowerBounds(frameTime))  {  score.Player2Scored();  if (score.isGameFinished())  {  FinishGame();  }  else  {  Scored(new Velocity(133, -93));  }  }   if (ball.IsOutOfUpperBounds(frameTime))  {  score.Player1Scored();  if (score.isGameFinished())  {  FinishGame();  }  else  {  Scored(new Velocity(133, 93));  }  }  }   private void Scored(Velocity newBallVelocity)  {  ball.center();  ball.setVelocity(newBallVelocity);  SoundManager.*playSound*(2, 1);  delayTime = 2;  }   private void FinishGame()  {  SoundManager.*playSound*(3, 1);  setState(*STATE\_PAUSE*);  AddToastToQueue(score.CreateWinnerBoard());  AddToastToQueue("Select Menu for a new game.");  ResetGame();  }   private void AdvanceBall(double frameTime)  {  ball.Move(frameTime);  battTop.Move(frameTime);  battBottom.Move(frameTime);  }   private void CheckCollision()  {  if (battBottom.CollidesWith(ball, frameTime))  {  ball.setPreviousLocation(frameTime);  ball.generateNewVelocityUp();  battBottom.setPreviousLocation(frameTime);  SoundManager.*playSound*(1, 1);  }  else if (battTop.CollidesWith(ball, frameTime))  {  ball.setPreviousLocation(frameTime);  ball.generateNewVelocityDown();  battTop.setPreviousLocation(frameTime);  SoundManager.*playSound*(1, 1);  }  }   public void setSurfaceSize(int width, int height)  {  synchronized (surfaceHolder)  {  canvasWidth = width;  canvasHeight = height;   backgroundImage = Bitmap.*createScaledBitmap*(backgroundImage, width, height, true);  ball.canvasChanges(width, height);  battBottom.canvasChanges(width, height);  battTop.canvasChanges(width, height);  ball.center();  SetInitialBattPosition();  }  }   public void setRunning(boolean isRunning)  {  this.isRunning = isRunning;  }   public void setState(int state)  {  this.currentState = state;  }   public void doStart0p()  {  synchronized (surfaceHolder)  {  previousState = *STATE\_RUNNING\_0P*;  setState(*STATE\_RUNNING\_0P*);  ResetGame();  }  }   private void ResetGame()  {  ball.center();  battTop.centerHorizontal();  battBottom.centerHorizontal();  setRunning(true);  ResetVelocities();  score.Reset();  delayTime = 2;  }   private void ResetVelocities()  {  battBottom.setVelocity(new Velocity(0, 0));  battTop.setVelocity(new Velocity(0, 0));  }   public void doStart1p()  {  synchronized (surfaceHolder)  {  previousState = *STATE\_RUNNING\_1P*;  setState(*STATE\_RUNNING\_1P*);  ResetGame();  }  }   public void doStart2p()  {  synchronized (surfaceHolder)  {  previousState = *STATE\_RUNNING\_2P*;  setState(*STATE\_RUNNING\_2P*);  ResetGame();  }  }   public void pause()  {  synchronized (surfaceHolder)  {  setState(*STATE\_PAUSE*);  }  }   public void unpause()  {  synchronized (surfaceHolder)  {  if (currentState == *STATE\_PAUSE*)  {  setState(previousState);  }  }  }   public void toggleDiagnosticInformation()  {  shouldDiagnosticInformation = !shouldDiagnosticInformation;  resumeGame();  }   public void setBattPosition(float xPosition1, float yPosition1, float xPosition2, float yPosition2)  {  if (currentState == *STATE\_RUNNING\_1P* || currentState == *STATE\_RUNNING\_2P*)  {  if (currentState == *STATE\_RUNNING\_1P*)  {  int battXPos = (int) xPosition1;  if (battXPos <= 0)  battXPos = 0;   battBottom.setxPosition(battXPos);  if (battBottom.IsMaximumRight(canvasWidth))  battBottom.SetMaximumRight(canvasWidth);  }  else if (currentState == *STATE\_RUNNING\_2P*)  {  if (yPosition1 < yPosition2)  {  SetBattsPosition((int) xPosition1, (int) xPosition2);  }  else  {  SetBattsPosition((int) xPosition2, (int) xPosition1);  }  }  }  }   private void SetBattsPosition(int xPosition1, int xPosition2)  {  battTop.setxPosition((int) xPosition1);  battBottom.setxPosition((int) xPosition2);  if (battTop.IsMaximumRight(canvasWidth))  battTop.SetMaximumRight(canvasWidth);  if (battBottom.IsMaximumRight(canvasWidth))  battBottom.SetMaximumRight(canvasWidth);  }   public void toggleSound()  {  SoundManager.*togglePlaySound*();  resumeGame();  }   public void resumeGame()  {  setState(previousState);  } } |

#### -REFERENCIAS

Patrick Kalkman. Androng, a pong clone for Android. Link: [http://www.codeproject.com/Articles/189515/Androng-a-Pong-clone-for-Android#Introduction0.](http://www.codeproject.com/Articles/189515/Androng-a-Pong-clone-for-Android%23Introduction0) Febrero 2016

**EJECUCIÓN DEL PROGRAMA EN ANDROID STUDIO**

|  |
| --- |
| **INICIO DEL JUEGO**  Patrón de fondo  Descripción generada automáticamente |
| **PRESENTACIÓN DEL JUEGO CORRIENDO**  Patrón de fondo  Descripción generada automáticamente |
| **FUNCIONALIDADES DEL JUEGO**  Patrón de fondo  Descripción generada automáticamente con confianza baja |

#### - CONCLUSIÓN

#### Para concluir, está practica fue la que más me costo en todos los sentidos, cada vez el grado de dificultad se fue agrandando bastante, aunque posiblemente no sea la mejor, me llevo varios conocimientos al tratar de programar todo esto, hasta el momento estoy satisfecho por todo lo adquirido en todas las practicas realizadas, sin más que agregar puedo decir que la practica fue finalizada de forma correcta.

#### -ANEXOS

ANEXO 1. Imagen de pelota de pong.

Animation frames of the Anrong ball

ANEXO 2. Imagen de Pong original

