Manlio Emiliano Terán Ramos
Paul Edward Castro Nava

Ing. Cibernética y Sistemas Computacionales.

# Lenguaje de Programación "MandroidPC"

TERÁN MANLIO, CASTRO PAUL

***Resumen*** **– se creó un nuevo lenguaje de programación orientado a Java Android SDK, con el fin de facilitar su acceso a la gente que no está familiarizada con este lenguaje. Gracias a esto, se reduce el tiempo de programación usando pocas líneas de código simplificando un trabajo más extenso. La finalidad del código está orientada a los video juegos para celulares (Gráficos), algunos ejemplos de gráficos que se implementaron fueron círculo, cuadrado y recta.**

## INTRODUCCION

El sistema operativo de celulares "Android" ha tenido un impacto muy fuerte en lo que es tecnología móvil. La gran parte del mercado móvil a nivel mundial lo tiene éste tipo de sistemas.

Debido a lo anterior, desarrollamos un nuevo lenguaje de programación básico mucho más sencillo de usar que el SDK Android, al cual llamamos "MandroidPC". El objetivo es inducir a la gente a éstos tipos de lenguajes improvisados para que se adentren al desarrollo de aplicaciones móviles y con esto lograr que en México haya más desarrolladores orientados a aplicaciones y Video Juegos móviles.

Independientemente de la competencia que haya entre los lenguajes de programación móviles, IOS y Android. El objetivo no es preferir uno, sino todo lo contrario, es adentrar a la gente a estos lenguajes de programación.

## METODOLOGIA

Se desarrolló en lenguaje de programación "C" un scanner que lea lo que se programe en el lenguaje de programación "MandroidPC" en un documento de texto, el cuál pasará por una técnica llamada "Parsing PDRC (Descendencia Recursiva por Código)". La cual localizará todas las instrucciones que reconozca como lenguaje de programación "MandroidPC" y lo interpretará a Lenguaje Java Android del SDK con un documento de texto de salida llamado "java.txt".

A la hora de codificar:

- Se implanta cada Producción como una rutina de código.
- Se emplea el propio Stack del Lenguaje (Llamado de Subrutinas o Procedimientos).
- Requiere BNF Extendido.
- Utilización de las funciones (Expect, CurrentToken, CurrentTokenInFirst).

Se implementaron Reglas PDRC:

Manlio Emiliano Terán Ramos
Paul Edward Castro Nava



## PDRC: Regla 1

- BNF
  - A::=B
- Traduce a
  - void A(void) {
    B();
    }

## PDRC: Regla 2

- BNF
  - B::=a
- Traduce a
  - void B(void) {
    Expect(a);
    }

## PDRC: Regla 3

- BNF
  - $A::=\alpha_1\alpha_2...\alpha_n$
- Traduce en
  - void A(void) {
    $\delta(\alpha_1)$;
    $\delta(\alpha_2)$;
    ...
    $\delta(\alpha_n)$;
    }
- Si
  - $\alpha_i \subset T \rightarrow \delta(\alpha_i) = Expect(\alpha_i)$;
  - $\alpha_i \subset N \rightarrow \delta(\alpha_i) = \alpha_i()$;

## PDRC: Regla 4

- BNF
  - B::=[α]
- Traduce a
  - Si α ⊂ T

```
void B(void) {
  if(CurrentToken(α))
    Expect(α)
}
```

✓Traduce a

❑Si α ⊂ N

```
void B(void) {
  if(CurrentTokenIn First(α))
    α();
}
```

## PDRC: Regla 5

- BNF
  - B::={α}
- Traduce a
  - Si α ⊂ T

```
void B(void) {
  while(CurrentToken(α))
    Expect(α)
}
```

✓Traduce a

❑Si α ⊂ N

```
void B(void) {
  while(CurrentTokenIn First(α))
    α();
}
```

## PDRC: Regla 6

- BNF
  - $A::= \alpha_1 \mid \alpha_2 \mid ... \mid \alpha_n$
- Traduce a
  - Si $\alpha_i \subset T$

```
void A(void) {
  if(CurrentToken(α1))
    Expect(α1)
  else{
    if(CurrentToken(α2))
      Expect(α2))
    else {...}
  }
}
```

✓Traduce a

❑Si $\alpha_i \subset N$

```
void B(void) {
  if(CurrentTokenIn First(α1))
    α1();
  else{
    if(CurrentTokenIn First(α2))
      α2();
    else {...}
  }
}
```

## PDRC: Regla 7

- Las reglas pueden mezclarse
  - Si hay varios símbolos o términos
    - A algunos términos les aplica una regla o varias,
    - Y en otros otra o varias

Al igual que las reglas, también se implementaron funciones ya diseñadas con anterioridad:

Manlio Emiliano Terán Ramos
Paul Edward Castro Nava



## Función CurrentToken

```
bool CurrentToken(int intSimboloT)
{
  if (pListaTokens_head->token_codigo == intSimboloT)
        return (TRUE);
  else
        return(FALSE);
}
```

## Función Expect

```
void Expect(int intSimboloT)
{
    if (CurrentToken(intSimboloT)
        pListaTokens_head =
            pListaTokens_head->sig;
    else
        printf("Error Sintáctico");
}
```

## Función CurrentTokenInFirst

```
bool CurrentTokenInFirst(int intSimboloNT)
{
    bool res=FALSE;
    switch (intSimboloNT)
    {
        case SIMB_NT_Operando:
            if (CurrentToken(SIMB_SAL_ID) res=TRUE;
            if (CurrentToken(SIMB_SAL_CONST) res=TRUE;
            if (CurrentToken(SIMB_SAL_EXP) res=TRUE;
            break;
        ...
    }
    return(res);
}
```

También se agregaron gramáticas del nuevo lenguaje de programación "MandroidPC" algunas de ellas son CÍRCULO, CUADRADO y RECTA. Se expresan de la siguiente manera:

# InstruccionCirculo

Se puede apreciar en la figura 1.

```
//InstruccionCirculo ::= "circulo" "("  Numero
// ":" Numero ":" Numero ":" ID ")"
void InstruccionCIRCULO()
{    Expect(SIMBOLO_TERMINAL_CIRCULO);
     Expect(SIMBOLO_TERMINAL_PARENTESIS_ABIERTO);
     Numero();
     Expect(SIMBOLO_TERMINAL_DOS_PUNTOS);
     Numero();
     Expect(SIMBOLO_TERMINAL_DOS_PUNTOS);
     Numero();
     Expect(SIMBOLO_TERMINAL_DOS_PUNTOS);
     Expect(SIMBOLO_TERMINAL_VARIABLE);
     Expect(SIMBOLO_TERMINAL_PARENTESIS_CERRADO);
}
```

Figura 1

# InstruccionRecta

Se puede apreciar en la figura 1.1.

```
//InstruccionRecta ::= "recta" "("  Numero ":"
//Numero ":" Numero ":" Nuemro ":" ID ")"
void InstruccionRECTA()
{
     Expect(SIMBOLO_TERMINAL_RECTA);
     Expect(SIMBOLO_TERMINAL_PARENTESIS_ABIERTO);
     Numero();
     Expect(SIMBOLO_TERMINAL_DOS_PUNTOS);
     Numero();
     Expect(SIMBOLO_TERMINAL_DOS_PUNTOS);
     Numero();
     Expect(SIMBOLO_TERMINAL_DOS_PUNTOS);
     Numero();
     Expect(SIMBOLO_TERMINAL_DOS_PUNTOS);
     Expect(SIMBOLO_TERMINAL_VARIABLE);
     Expect(SIMBOLO_TERMINAL_PARENTESIS_CERRADO);
}
```

Figura 1.1

# InstruccionCuadrado

Se puede apreciar en la figura 1.2.

```
//InstruccionCUADRADO ::= "CUADRADO" "("  Numero ":
//Numero ":" Numero ":" Numero ":" ID ")"
void InstruccionCUADRADO()
{
    Expect(SIMBOLO_TERMINAL_CUADRADO);
    Expect(SIMBOLO_TERMINAL_PARENTESIS_ABIERTO);
    Numero();
    Expect(SIMBOLO_TERMINAL_DOS_PUNTOS);
    Numero();
    Expect(SIMBOLO_TERMINAL_DOS_PUNTOS);
    Numero();
    Expect(SIMBOLO_TERMINAL_DOS_PUNTOS);
    Numero();
    Expect(SIMBOLO_TERMINAL_DOS_PUNTOS);
    Expect(SIMBOLO_TERMINAL_VARIABLE);
    Expect(SIMBOLO_TERMINAL_PARENTESIS_CERRADO);
}
```

Figura 1.2

**CONCEPTOS BASICOS**

Algunas de las instrucciones que implementamos en el lenguaje de programación “MandroidPC” son:

CIRCULO(50:50:70 paint);

RECTA(500:100:50:50:paint);

CUADRADO(125:125:70:70:paint);

Estas instrucciones tienen números dentro de ellas. Las cuales son el tamaño y las dimensiones de cada dibujo.

Por último tenemos una variable llamada paint, que será la variable que incluye al objeto “Paint” de la clase “Canvas” de Java Android SDK.

Para poder iniciar el lenguaje de programación “MandroidPC” se necesita la instrucción siguiente:

INICIAR{

}

Dentro de la cual se necesitará implementar las instrucciones ya conocidas:

INICIAR{

CIRCULO(50:50:70 paint);

RECTA(500:100:50:50:paint);

CUADRADO(125:125:70:70:paint);

}

La salida que generará esto estará dentro del documento de texto “java.txt” y quedará algo parecido a esto:

```
import android.app.Activity;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Paint.Style;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
public class MainActivity extends Activity {
class RenderView extends View {
Paint paint;
public RenderView(Context context) {
super(context);
paint = new Paint();
}
protected void onDraw(Canvas canvas) {
paint.setColor(Color.RED);
canvas.drawLine(50,50,50,50,paint);
canvas.drawCircle(100,100,50,paint);
canvas.drawCircle(50,50,70,paint);
canvas.drawRect(250,250,100,100,paint);
canvas.drawLine(500,100,50,50,paint);
canvas.drawRect(125,125,70,70,paint);
invalidate();
}
}
@Override
public void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
requestWindowFeature(Window.FEATURE_NO_
TITLE);
getWindow().setFlags(WindowManager.LayoutPa
rams.FLAG_FULLSCREEN,
```

```
WindowManager.LayoutParams.FLAG_FULLSC
REEN);
setContentView(new RenderView(this));
}
}
```

El cuál es el lenguaje de programación de Java Android SDK y al correrlo se obtienen las tres figuras.

Los “imports” de la parte superior del programa generado son llamados paquetes que en java es un contenedor de clases que permite agrupar las distintas partes de un programa cuya funcionalidad tienen elementos comunes.

## RESULTADOS

Se puede apreciar en la figura 1.3.

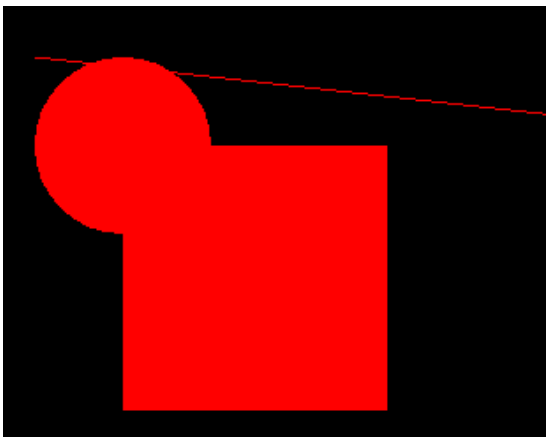

Figura 1.3

## CONCLUSIONES

Como pudimos ver el resultado, el lenguaje de programación de Java Android SDK se simplifico de manera drástica gracias al lenguaje de programación “MandroidPC”, logrando hacer con 5 líneas de código lo mismo que las mostradas con anterioridad.

Con todo esto se puede tener aplicaciones y video juegos móviles de manera más sencilla, logrando que las personas tengan más acceso a éste sistema operativo.

## REFERENCIAS

1 http://infomundo.org/informatica/android-ya-es-la-referencia-en-el-mercado-pese-a-la-llegada-de-ios-8/

2 http://www.dirinfo.unsl.edu.ar/compi/teorias/PDR.pdf

3 Apuntes de la materia Compiladores y Traductores, Jorge Kashiwamoto.