Universidad de San Carlos de Guatemala
Centro Universitario de Occidente
División de Ciencias de la Ingeniería

“IDE Y LENGUAJE KOKY
MANUAL DE USUARIO”

Versión 2.0

INGENIERÍA EN CIENCIAS Y SISTEMAS

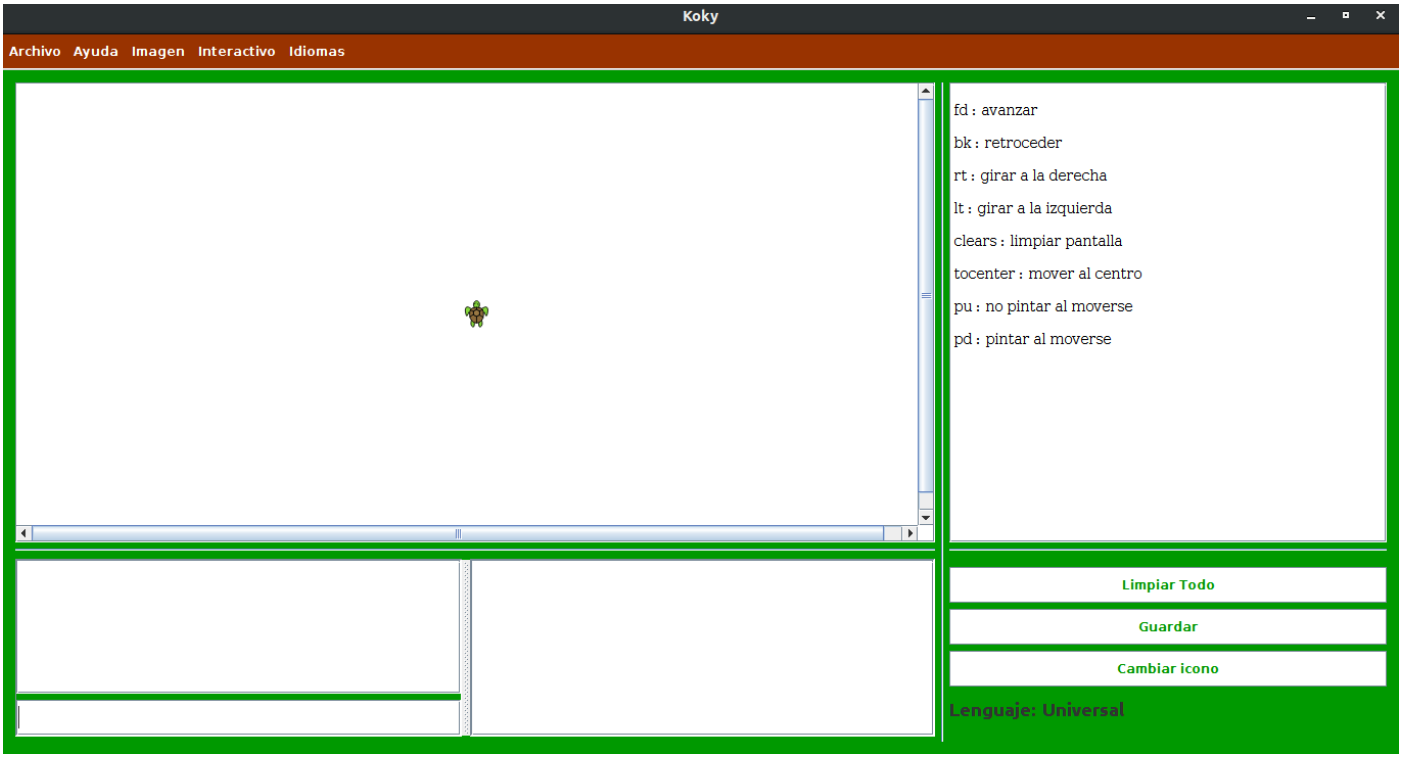
Quetzaltenango, Septiembre, 2019

1. Introducción

El lenguaje e IDE koky esta pensado y diseñado para facilitar la introducción de niños al área de programación, esto por medio de instrucciones sencillas y no ambiguas que se traducen en acciones realizadas por un personaje lo cual da una idea divertida de la programación siendo parecida mas a un juego.

2. Inicio (Instalación e ingreso)

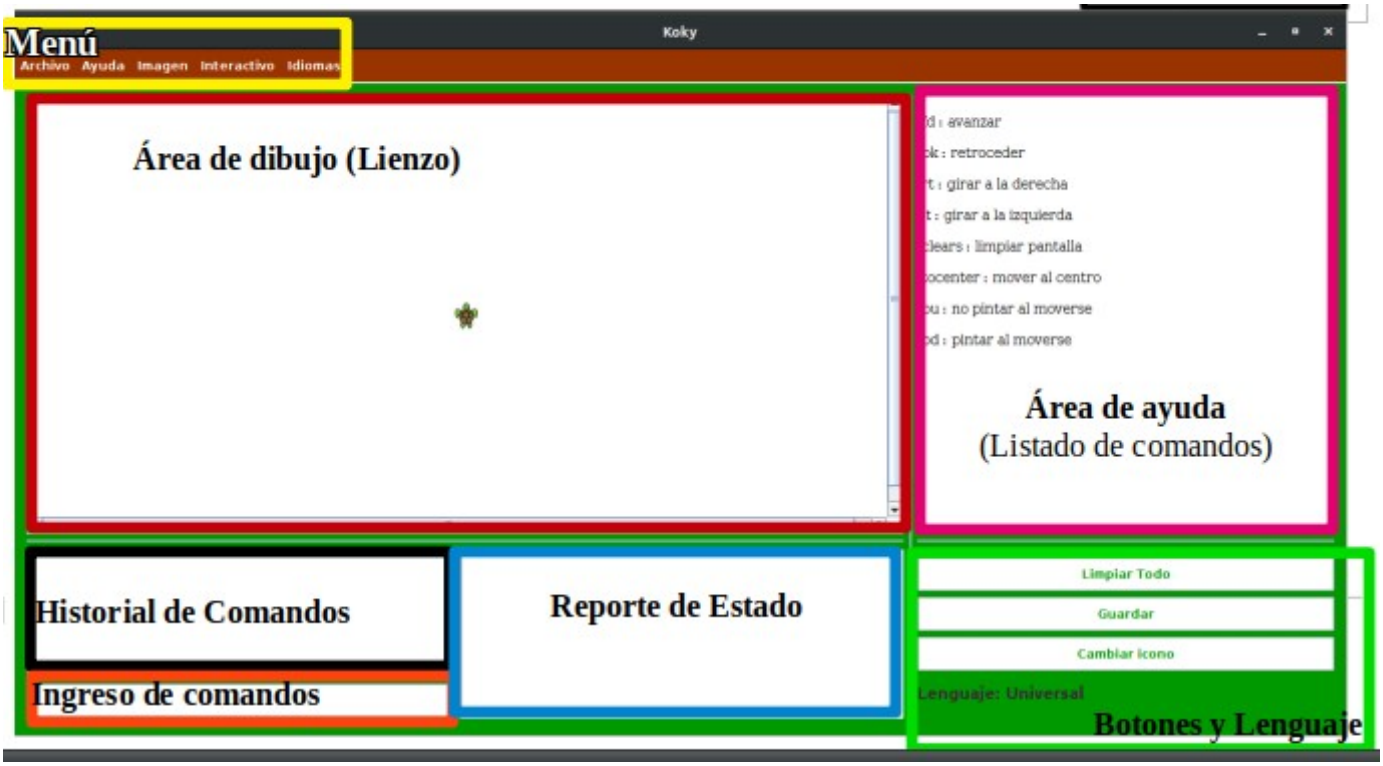
No necesita el inicio de sesiones, la siguiente pagina es la principal.



3. Áreas de Trabajo

La pagina principal cuenta con las siguiente áreas de trabajo:

- Menú
- Área de dibujo
- Botones
- Ayuda (descripción)
- Historial de comandos
- Ingreso de Comandos
- Reporte de estado



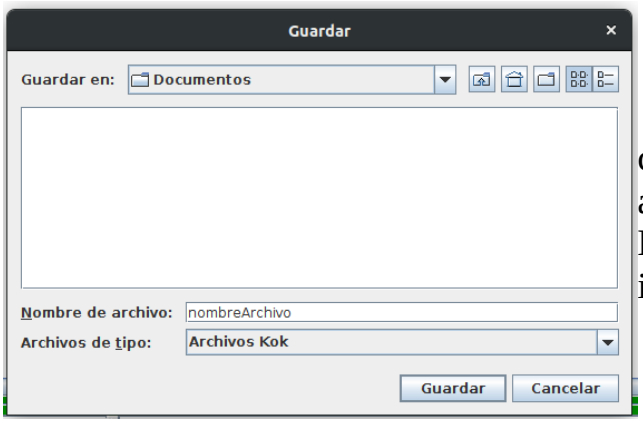
3.1 Menú

Cuenta con 3 opciones Archivo, Ayuda y guardar, al dar click a alguna es ellas despliega ciertas opciones.

- Archivo
  - Guardar Instrucciones
  - Abrir el editor de texto
- Ayuda
  - Instrucciones
  - Consultar color
  - Acerca de
- Imagen
  - Guardar imagen
  - Cambiar tamaño imagen

- Interactivo
  - ¡Tomar un reto!
  - Historial de retos
  - Dibujos predefinidos
- Idiomas

3.1.1 Archivo



3.1.1.1 Guardar Instrucciones

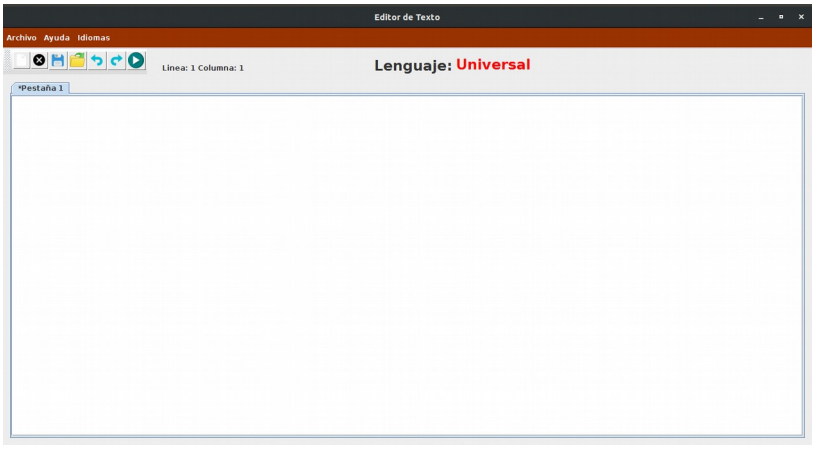
Abre una ventana de dialogo la cual tiene como función el elegir la ubicación del archivo y el nombre con el cual se guardara. El archivo guarda el historial de comandos ingresados.

3.1.1.2 Abrir el Editor de Texto

Abre una nueva ventana con un sub-menu y las funciones básicas de un editor de texto.

Las opciones del editor de texto son:

- Crear pestaña
- Eliminar pestaña
- Guardar archivo
- Abrir archivo
- Deshacer
- Rehacer
- Ejecutar



3.1.2 Ayuda

fd : avanzar

bk : retroceder

rt : girar a la derecha

lt : girar a la izquierda

clears : limpiar pantalla

tocenter : mover al centro

pu : no pintar al moverse

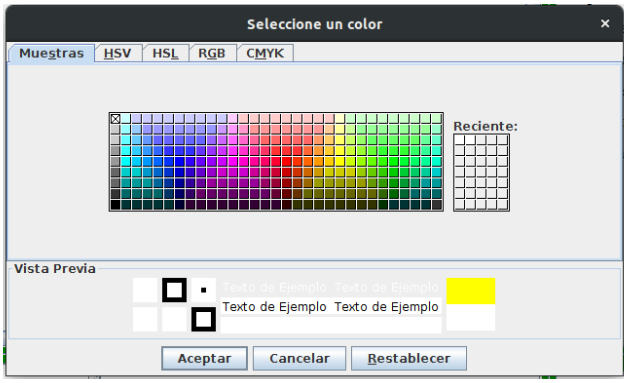
pd : pintar al moverse

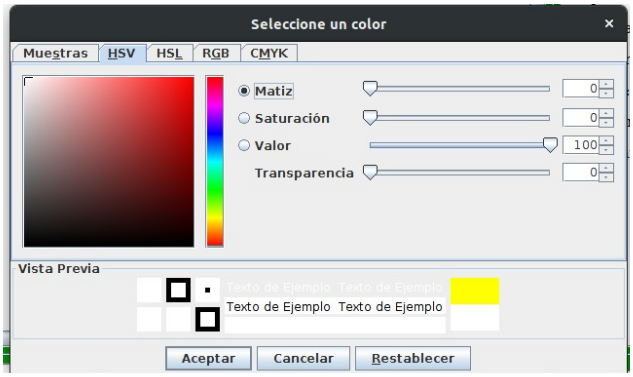
3.1.2.1 Instrucciones

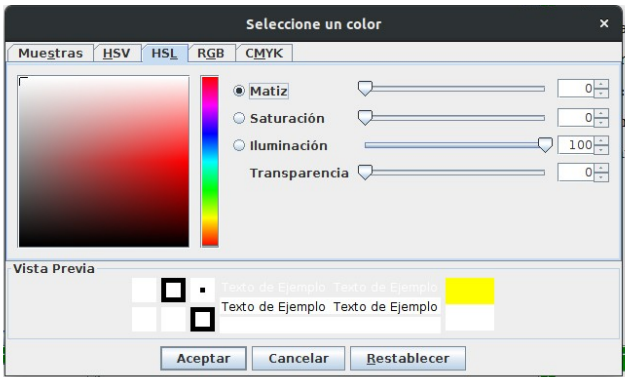
Ventana con el listado de todos los comandos junto con una descripción de lo que hacen y ejemplos.

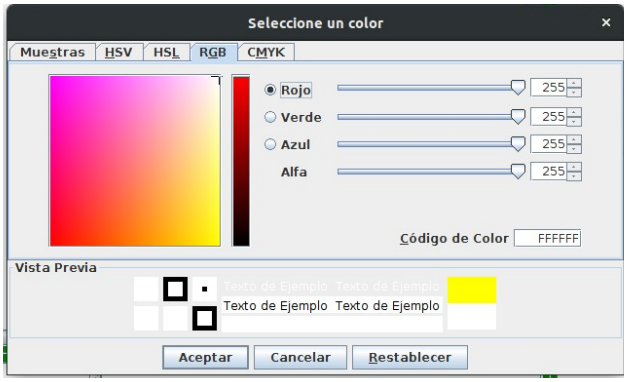
3.1.2.2 Consultar color

Despliega ventana con una paleta de colores para elegir, al seleccionar una y aceptar el color seleccionado se ingresa de manera automática al área de ingreso de comandos.









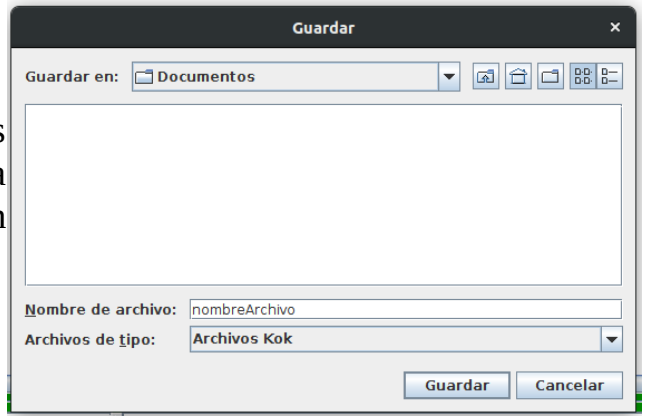
3.1.2.3 Acerca de

Da información básica de la aplicaciones versión, creadores y fecha de lanzamiento.

3.1.3 Imagen

3.1.3.1 Guardar imagen

Esta función guarda la imagen generada por los comandos en un archivo .jpg para esto abre una ventana para seleccionar la ubicación y agregarle un nombre.



3.1.3.2 Cambiar tamaño imagen

Da un mensaje para preguntar si el usuario acepta eliminar su progreso antes de generar otra ventana que pide el tamaño deseado de la imagen siendo el mínimo (900,450)

3.1.4 Interactivo

3.1.4.1 ¡Toma un reto!

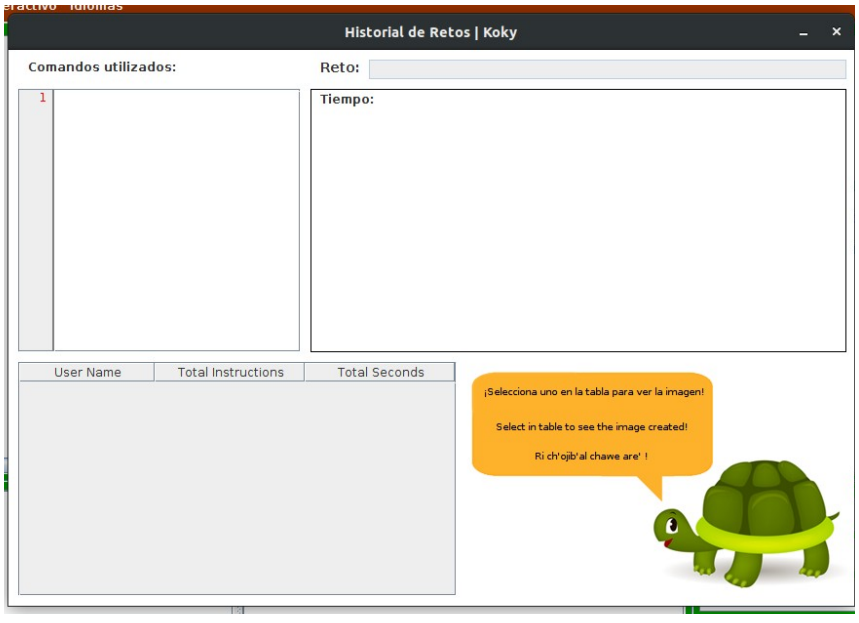
Al seleccionar esta opción nos pide confirmar el eliminar la imagen en pantalla, seguido nos pide un apodo o nombre para iniciar el reto. Nos muestra una ventana informando el reto que toco.

Los posibles retos son:

- Crea un triangulo isósceles, escaleno y equilátero.
- Crea una flor utilizando círculos.
- Deberás dibujar una casa que tenga una puerta y una ventana.
- ¡Dibuja una mariposa!
- Intenta escribir los números de 1 al 5.
- Dibuja un carita feliz.
- Intenta dibujar dos montañas y un sol.
- Dibuja un carro de color rojo.
- Algo fácil: Escribe tu nombre!
- Dibuja una cancha de fútbol.

3.1.4.2 Historial de retos

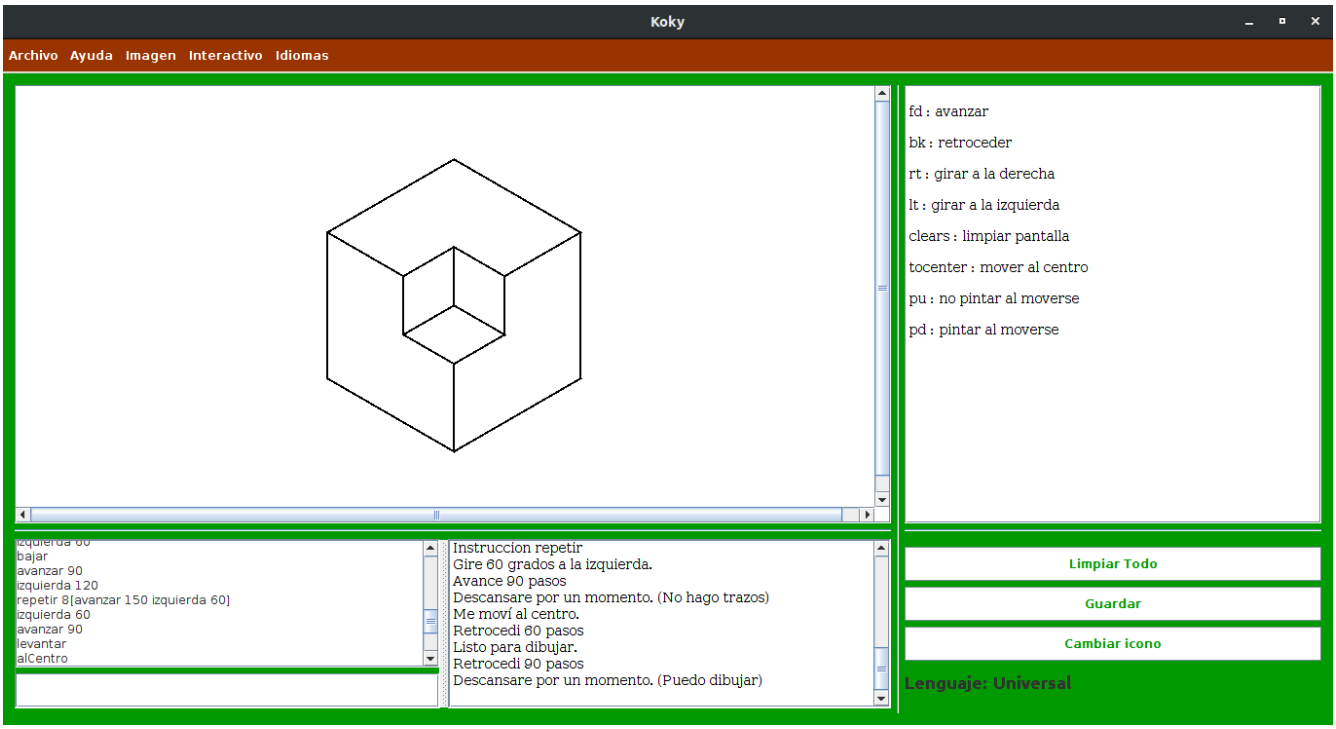
Despliega una pantalla que según los retos realizados hasta el momento selecciona algún reto esto acción carga en la tabla todos los usuarios que han realizado el reto y al seleccionar alguno se muestra los comandos que utilizo, el dibujo y el tiempo que tardo en realizarlo.



3.1.4.2 Dibujos predeterminados

Despliega una opción con 6 figuras, al seleccionar una de ellas automáticamente se dibujaran en el área de dibujo. Las figuras son las siguientes:

- Cuadrado
- Triangulo
- Circulo
- Pentágono
- Estrella
- Cubo



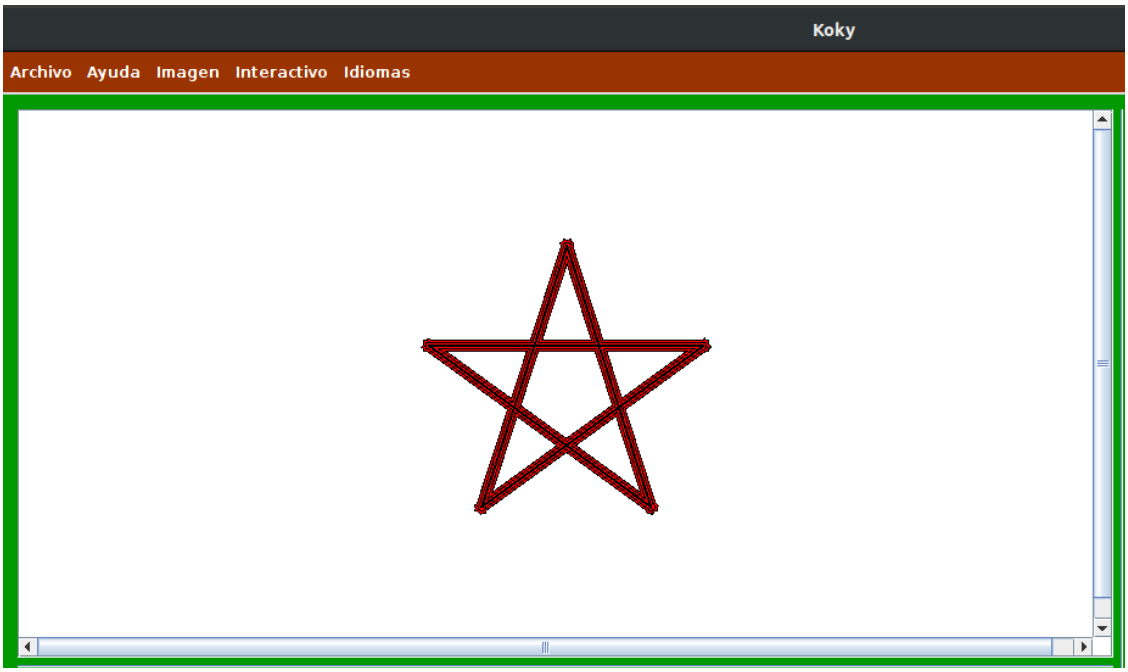
3.1.5 Idiomas

Despliega 4 posibilidades para configurar el lenguaje en el cual se desea ingresar los comandos.

- Todos
- Español
- Ingles
- Kiche

3.2 Área de dibujo

Es un lienzo blanco con un personaje el cual nos sirve como referencia para saber donde esta el pincel de pintura, este se ira moviendo y realizando acciones dependiendo de las instrucciones que se le den desde el área de comandos, de esta forma se dibujaran distintos diseños con colores y tamaños. Tiene por defecto un pincel tamaño 2, color negro y la figura de una tortuga centrada según sea el tamaño del lienzo tanto vertical como horizontal.



3.3 Ingreso de comandos

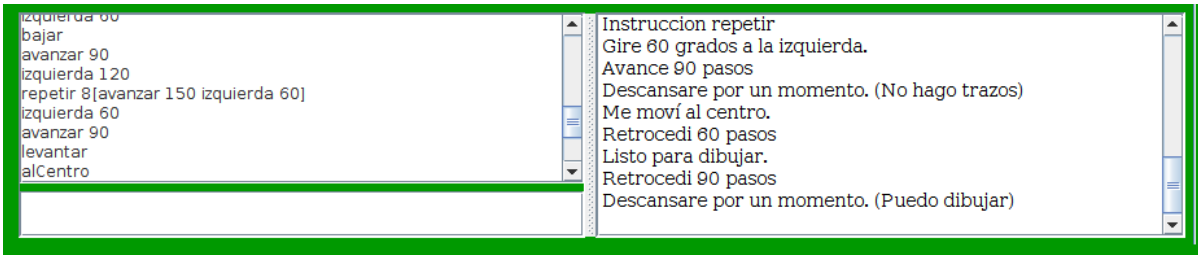
Pequeña área donde se le pueden dar instrucciones al personaje desde formato de comandos, no deben de ser ambiguos para que la tortuga pueda entender y seguirlas, esta área cuanta con una memoria la cual al presionar la flecha para arriba del teclado regresa al comando ingresado anterior.

3.4 Historial de comandos

Como su nombre lo dice, es una lista con todos los comandos ingresados hasta el momento; conforme se sigan ingresando comandos esta área se seguirá llenando tanto de los comandos bien ingresados como de los que no logre reconocer.

3.5 Reporte de Estado

Se encarga de informar al usuario lo que la tortuga esta haciendo en el lienzo, muestra información de sus movimiento, de las cosas que no entiende o mal ingresadas. Para llevar un control de los pasos que la tortuga da.



3.6 Ayuda

En esta área se listan los comandos validos según sea el idioma seleccionado.

3.7 Botones

Este sector cuenta con 3 botones y un texto que informa sobre el lenguaje seleccionado actualmente.

3.7.1 Limpiar

Limpia el lienzo, el historial de comandos y el reporte de estado, así también mueve al centro del lienzo a la tortuga.

3.7.2 Guardar

Abre una ventana de dialogo la cual tiene como función el elegir la ubicación del archivo y el nombre con el cual se guardara. El archivo guarda el historial de comandos ingresados.

3.7.3 Cambiar icono

Abre una ventana con diferentes estilos de tortugas para elegir el diseño que mas nos parezca, cuanta con 7 estilos y una opción para elegir de manera aleatoria. El cambio de tortuga se vera reflejado en el área de dibujo.

4. Comandos

4.1 Ingles

4.1.1 FORWARD (fd)

Sintaxis: **forward *n***

Donde ***n*** significa el número de pasos a avanzar hacia delante.

4.1.2 BACKWARD (bk)

Sintaxis: **backward *n***

Donde ***n*** significa el número de pasos a avanzar hacia atrás.

4.1.3 RIGHT (rt)

Sintaxis: **right *n***

Donde ***n*** significa el número de grados a girar hacia la derecha.

4.1.4 LEFT (lt)

Sintaxis: **left *n***

Donde ***n*** significa el número de grados a girar hacia la izquierda.

4.1.5 CLEARS (cs)

Sintaxis: **clears**

Limpia la pantalla y mueve la tortuga al centro.

4.1.6 PENUP (pu)

Sintaxis: **penup**

Hace que se levante la pluma de la tortuga y no genere trazos al moverse. Cancela la instrucción PenDown

4.1.7 PENDOWN (pd)

Sintaxis: **pendown**

Hace que se baje la pluma de la tortuga para generar trazos al moverse y es el comportamiento por defecto del cursor. Cancela la instrucción PenUp.

4.1.8 TOCENTER (cr)

Sintaxis: **tocenter**

Hace que se la tortuga se mueva al centro de la pantalla.

4.1.9 COLOR

Sintaxis: **color *n***

color hexa

Hace que el color de la pluma cambie. el número ***n*** debe estar entre 0 y 9, y representa uno de los 10 colores predefinidos. (queda a discreción del programador especificarlos) Si se desea usar algún color diferente a los existentes, el usuario puede definir un color usando un número hexadecimal de 6 dígitos. p.e. #A9F033

4.1.10 POSITIONXY (posxy)

Sintaxis: **positionxy *x,y***

Mueve la tortuga a la posición dada por los parámetros ***x*** y ***y.***

4.1.11 POSITIONX (posx)

Sintaxis: **positionx *x***

Mueve la tortuga sobre el eje ***x*** a la posición dada por el parámetro ***x.*** La posición sobre el eje ***y*** no debe cambiar.

4.1.12 POSITIONY (posy)

Sintaxis: **positiony *y***

Mueve la tortuga sobre el eje ***y*** a la posición dada por el parámetro ***y.*** La posición sobre el eje ***x*** no debe cambiar.

4.1.13 HIDETURTLE (ht)

Sintaxis: **hideturtle**

Oculta la tortuga de la pantalla. Cancela la instrucción ShowTurtle.

4.1.14 SHOWTURTLE (st)

Sintaxis: **showturtle**

Muestra la tortuga de la pantalla y es el comportamiento por defecto. Cancela la instrucción HideTurtle.

4.1.15 TOERASE (te)

Sintaxis: **toerase**

Activa el borrador, con el cual a cada paso que de la tortuga borra algún trazo existente. Cancela la instrucción ToDraw

4.1.16 TODRAW (td)

Sintaxis: **todraw**

Cancela el borrador, con el cual a cada paso que de la tortuga generará trazos. Es el comportamiento por defecto y cancela la instrucción ToErase.

4.1.17 REPEAT

Sintaxis: **repeat *n* [<secuencia de comandos a repetir>]**

Repite una cantidad ***n*** la secuencia de instrucciones dada entre los corchetes.

4.1.18 WIDTH (wd)

Sintaxis: **width *n***

Donde ***n*** significa el tamaño del pincel, debe de estar entre 1 y 15.

4.1.19 VOID

Sintaxis: **void :<Nombre que llevara la función> (:parametro1,:parametro2,…:parametroN)**

[< Secuencia de comandos de la función>]

Ejemplo:

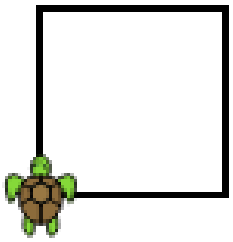
void :cuadrado(:ancho) [repeat 4 [fd :ancho rt 90]]

void :avanzar(:pasos) [fd :pasos]

Nota: lo que esta entre paréntesis son los parámetros, los cuales servirán para que puedan ir variando los números que se asignaran a instrucciones dentro de cada función. Estos parámetros tendrán un valor o numero asignado en el momento que se llame a esa función es decir: llamar :cuadrado(50) de esta forma lo que hará es un cuadrado de 50 pixeles, si se llama de esta otra forma: llamar :cuadrado(100) lo que hará es que se mostrara un cuadrado de 100 pixeles. En la parte del comando CALL se explica como llamar a las funciones mas detalladamente.

```
void :cuadrado(:ancho) [  repeat 4 [fd :ancho rt 90 ] ]
call :cuadrado(60)
```

Resultado de esta operación: cuadrado con lados de 60 pixeles



4.1.20 CALL

Sintaxis: **call :<nombre de la funcion previamente declarada> (:parametro1,:parametro2…:parametroN)**
Ejemplo:

call :cuadrado(200)

Lo que hace esto es que esta llamando a la función :cuadrado y asignando un valor de 200 al único parámetro que existe en esa función que tiene que estar previamente declarada de forma que acepte un parámetro.

Otro ejemplo:

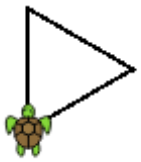
call :hola(300,400+100)

Lo que hace esto es que esta llamando a la función :hola que acepta dos parámetros que tendrán valores de 300 el primero y 500 (resultado de sumar 400+100) para el segundo parámetro, La función :hola tiene que aceptar dos parámetros.

Ejemplo:

```
void :triangulo(:lado) [  repeat 3 [fd :lado rt 120 ] ]
call :triangulo(60)
```

Resultado de la operación anterior: triangulo con lados de 60 pixeles



Kiche

4.2.1 B'INIK (bk)

Sintaxis: **b'inik *n***

Donde ***n*** significa el número de pasos a avanzar hacia delante.

4.2.2 TZALIJIK (tk)

Sintaxis: **tzalijik *n***

Donde ***n*** significa el número de pasos a avanzar hacia atrás.

4.2.3 K'IQ'AB' (kb)

Sintaxis: **k’iq’ab’ *n***

Donde ***n*** significa el número de grados a girar hacia la derecha.

4.2.4 MOXQ'AB (mb)

Sintaxis: **moxq’ab *n***

Donde ***n*** significa el número de grados a girar hacia la izquierda.

4.2.5 SU'NIK (sk)

Sintaxis: **su’nik**

Limpia la pantalla y mueve la tortuga al centro.

4.2.6 WA'LIJIK (wk)

Sintaxis: **wa’lijik**

Hace que se levante la pluma de la tortuga y no genere trazos al moverse. Cancela la instrucción qajik

4.2.7 QAJIK (qk)

Sintaxis: **qajik**

Hace que se baje la pluma de la tortuga para generar trazos al moverse y es el comportamiento por defecto del cursor. Cancela la instrucción wa’lijik.

4.2.8 OJPANIK'AJ (oj)

Sintaxis: **ojpanik’aj**

Hace que se la tortuga se mueva al centro de la pantalla.

4.2.9 KAYB'ALIL

Sintaxis: **kayb’alil *n***

color hexa

Hace que el color de la pluma cambie. el número ***n*** debe estar entre 0 y 9, y representa uno de los 10 colores predefinidos. (queda a discreción del programador especificarlos) Si se desea usar algún color diferente a los existentes, el usuario puede definir un color usando un número hexadecimal de 6 dígitos. p.e. #A9F033

4.2.10 KRIQTAJXY (kjxy)

Sintaxis: **kriqtajxy *x,y***

Mueve la tortuga a la posición dada por los parámetros ***x*** y ***y.***

4.2.11 KRIQTAJX (kjx)

Sintaxis: **kriqtajx *x***

Mueve la tortuga sobre el eje ***x*** a la posición dada por el parámetro ***x.*** La posición sobre el eje ***y*** no debe cambiar.

4.2.12 KRIQTAJY (kjy)

Sintaxis: **kriqtajy *y***

Mueve la tortuga sobre el eje ***y*** a la posición dada por el parámetro ***y.*** La posición sobre el eje ***x*** no debe cambiar.

4.2.13 UK'U'IKKOK (uk)

Sintaxis: **uk’u’ikkok**

Oculta la tortuga de la pantalla. Cancela la instrucción rilikkok.

4.2.14 RILIKKOK (rk)

Sintaxis: **rilikkok**

Muestra la tortuga de la pantalla y es el comportamiento por defecto. Cancela la instrucción uk’u’ikkok.

4.2.15 CHUPUB'AL (cl)

Sintaxis: **chupub’al**

Activa el borrador, con el cual a cada paso que de la tortuga borra algún trazo existente. Cancela la instrucción utzjuch’unik

4.2.16 UTZJUCH'UNIK (uj)

Sintaxis: **utzjuch'unik**

Cancela el borrador, con el cual a cada paso que de la tortuga generará trazos. Es el comportamiento por defecto y cancela la instrucción chupub'al.

4.2.17 JUTIJCHIK

Sintaxis: **jutijchik *n* [<secuencia de comandos a repetir>]**

Repite una cantidad ***n*** la secuencia de instrucciones dada entre los corchetes.

4.2.18 UWACH (uw)

Sintaxis: **uwach *n***

Donde ***n*** significa el tamaño del pincel, debe de estar entre 1 y 15.

4.2.19 KJEQE'IK

Sintaxis: **kjeque'ik :<nombre de la función> (:parametro1,:parametro2, …:parametroN) [<secuencia de comandos de la función>]**

Ejemplo:

```
kjeqe'ik :dibujarLinea ()  [ b'inik 90]
sik'im :dibujarLinea()
```

Resultado de esta operación: mover a la tortuga hacia adelante 90 pixeles

4.2.20 SIK'IM

Sintaxis: **sik'im :<nombre de la función> (:parametro1,:parametro2, …:parametroN)**

Ejemplo:

```
kjeqe'ik :dibujarLinea ()  [ b'inik 90]
sik'im :dibujarLinea()
```

Resultado de esta operación: mover a la tortuga hacia adelante 90 pixeles

4.3 Español

4.3.1 AVANZAR (ar)

Sintaxis: **avanzar *n***

Donde ***n*** significa el número de pasos a avanzar hacia delante.

4.3.2 RETROCEDER (rr)

Sintaxis: **retroceder *n***

Donde ***n*** significa el número de pasos a avanzar hacia atrás.

4.3.3 DERECHA (da)

Sintaxis: **derecha *n***

Donde ***n*** significa el número de grados a girar hacia la derecha.

4.3.4 IZQUIERDA (ia)

Sintaxis: **izquierda *n***

Donde ***n*** significa el número de grados a girar hacia la izquierda.

4.3.5 LIMPIAR (lr)

Sintaxis: **limpiar**

Limpia la pantalla y mueve la tortuga al centro.

4.3.6 LEVANTAR (lv)

Sintaxis: **levantar**

Hace que se levante la pluma de la tortuga y no genere trazos al moverse. Cancela la instrucción bajar

4.3.7 BAJAR (br)

Sintaxis: **bajar**

Hace que se baje la pluma de la tortuga para generar trazos al moverse y es el comportamiento por defecto del cursor. Cancela la instrucción levantar.

4.3.8 ALCENTRO (cr)

Sintaxis: **alcentro**

Hace que se la tortuga se mueva al centro de la pantalla.

4.3.9 COLOR

Sintaxis: **color *n***

color hexa

Hace que el color de la pluma cambie. el número ***n*** debe estar entre 0 y 9, y representa uno de los 10 colores predefinidos. (queda a discreción del programador especificarlos) Si se desea usar algún color diferente a los existentes, el usuario puede definir un color usando un número hexadecimal de 6 dígitos. p.e. #A9F033

4.3.10 POSICIONXY (posxy)

Sintaxis: **posicionxy *x,y***

Mueve la tortuga a la posición dada por los parámetros ***x*** y ***y.***

4.3.11 POSICIONX (posx)

Sintaxis: **posicionx *x***

Mueve la tortuga sobre el eje ***x*** a la posición dada por el parámetro ***x.*** La posición sobre el eje ***y*** no debe cambiar.

4.3.12 POSICIONY (posy)

Sintaxis: **posiciony *y***

Mueve la tortuga sobre el eje ***y*** a la posición dada por el parámetro ***y.*** La posición sobre el eje ***x*** no debe cambiar.

4.3.13 TAPARTORTUGA (tt)

Sintaxis: **tapartortuga**

Oculta la tortuga de la pantalla. Cancela la instrucción vertortuga.

4.3.14 VERTORTUGA (vt)

Sintaxis: **vertortuga**

Muestra la tortuga de la pantalla y es el comportamiento por defecto. Cancela la instrucción tapartortuga.

4.3.15 BORRADOR (bo)

Sintaxis: **borrador**

Activa el borrador, con el cual a cada paso que de la tortuga borra algún trazo existente. Cancela la instrucción dibujar

4.3.16 DIBUJAR (di)

Sintaxis: **dibujar**

Cancela el borrador, con el cual a cada paso que de la tortuga generará trazos. Es el comportamiento por defecto y cancela la instrucción borrador.

4.3.17 REPETIR

Sintaxis: **repetir *n* [<secuencia de comandos a repetir>]**

Repite una cantidad ***n*** la secuencia de instrucciones dada entre los corchetes.

4.3.18 ANCHO (an)

Sintaxis: **ancho *n***

Donde ***n*** significa el tamaño del pincel, debe de estar entre 1 y 15.

4.3.19 PROCESO

Sintaxis: **proceso :<Nombre que llevara la función> (:parametro1,:parametro2,…:parametroN)**

[< Secuencia de comandos de la función>]

Ejemplo:

proceso :cuadrado(:ancho) [repetir 4 [avanzar :ancho derecha 90]]

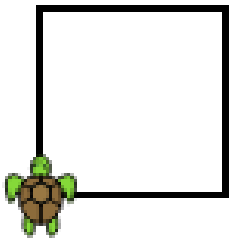
proceso :avanzar(:pasos) [avanzar :pasos]

Nota: lo que esta entre paréntesis son los parámetros, los cuales servirán para que puedan ir variando los números que se asignaran a instrucciones dentro de cada función. Estos parámetros tendrán un valor o numero asignado en el momento que se llame a esa función es decir: llamar :cuadrado(50) de esta forma lo que hará es un cuadrado de 50 pixeles, si se llama de esta otra forma: llamar :cuadrado(100) lo que hará es que se mostrara un cuadrado de 100 pixeles. En la parte del comando CALL se explica como llamar a las funciones mas detalladamente.

Ejemplo:

```
proceso :cuadrado(:ancho) [  repetir 4 [ avanzar :ancho derecha 90 ] ]
llamar :cuadrado(60)
```

Resultado de esta operación: cuadrado con lados de 60 pixeles

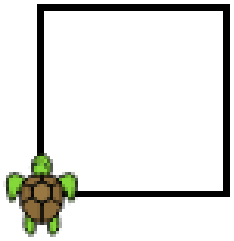


4.3.20 LLAMAR

Sintaxis: **llamar :<nombre de la función previamente declarada> (:parametro1,:parametro2…:parametroN)**
Ejemplo:

llamar :cuadrado(200)

Lo que hace esto es que esta llamando a la función :cuadrado y asignando un valor de 200 al único parámetro que existe en esa función que tiene que estar previamente declarada de forma que acepte un parámetro.



Otro ejemplo:

llamar :hola(300,400+100)

Lo que hace esto es que esta llamando a la función :hola que acepta dos parámetros que tendrán valores de 300 el primero y 500 (resultado de sumar 400+100) para el segundo parámetro, La función :hola tiene que aceptar dos parámetros.

Ejemplo:

```
proceso :triangulo(:lado) [  repetir 3 [ avanzar :lado derecha 120 ] ]
llamar :triangulo(30+30)
```

Resultado de la operación anterior: triangulo con lados de 60 pixeles