Universidad Autónoma Metropolitana
Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Reporte de Proyecto Terminal

Aproximación de la técnica de repixelización de arte pixelado

José Ricardo Silva Guerrero
Matrícula: 209200065

Trimestre: 13I
Abril 2013

Asesor: Dr. Risto Fermín Rangel Kuoppa
Profesor titular
Departamento de Sistemas

Agradecimientos

A mi familia.
Por haberme apoyado en todo momento, por sus consejos, sus valores, por la motivación constante que me ha permitido ser una persona de bien.

A Guadalupe.
Por los ejemplos de perseverancia y constancia que la caracterizan, que me ha infundado siempre y por el valor mostrado para salir adelante, pero más que nada, por su amor.

A mi asesor.
Por aceptarme para realizar este Proyecto Terminal bajo su dirección. Su confianza en mi trabajo y su capacidad para guiar mis ideas ha sido un aporte invaluable, no solamente en el desarrollo de este Proyecto Terminal, sino también en mi formación como Ingeniero en Computación.

Todo este trabajo ha sido posible gracias a ellos.

Índice General

Introducción

El *Pixel Art* es una expresión de arte de manera digital, donde los detalles de las imágenes son cuidadosamente representados al nivel de un píxel.

La presente propuesta contempla un interesante reto: ¿Es posible procesar un *sprite*[1] extraído de una fuente de poca calidad, para convertirlo en una representación de vectores independientes de la resolución?

En la década de los 90's se prescindía del potencial computacional con el que contamos hoy en día, lo cual dio lugar a que las imágenes tanto de PC, como de muchos videojuegos y algunos dispositivos móviles fueran representadas mediante *Pixel Art*.

De hecho, algunas imágenes *Pixel Art* han sido tan representativas de la técnica, que se han convertido en iconos culturales y son reconocidas instantáneamente por generaciones: *Bomberman*®, *Doom*™, *Space Invaders*™, *Super Mario World*™ (Imagen 1), etc.

Imagen 1. *Sprite* de *Super Mario World*™ (© Nintendo Co., Ltd.).

[1] *Sprite*: Serie de imágenes almacenadas dentro de un mismo archivo y que representan a un objeto en diferentes posiciones (movimiento).

Justificación

Anteriormente se prescindía del potencial computacional con el que contamos hoy en día, lo cual dio lugar a que toda una generación de imágenes fuera representada mediante *Pixel Art*. Debido a esto, en el campo del procesamiento digital de imágenes existe una necesidad de procesar esas imágenes para obtener imágenes de mayor calidad.

Como se describió en la sección anterior este proyecto terminal se enfoca en procesar un *sprite* extraído de una fuente de poca calidad, para convertirlo en una representación de vectores independientes de la resolución.

Actualmente, existen algoritmos (*2xSaI* **[2]**, *Scale2x* **[3]**, *hq2x/hq3x/hq4x* **[4]**, etc.) que resuelven este tipo de problemas mediante diferentes técnicas (interpolación[2], segmentación[3], vectorización, etc.), pero, en particular, el algoritmo que se plantea realizar no se encuentra disponible como una opción de software libre.

Por ello, tanto la presente propuesta, como el proyecto terminal estarán disponibles para todo el público.

Finalmente, se obtendrá un algoritmo para convertir imágenes *Pixel Art* en una representación de vectores independientes de la resolución.

Objetivo General

El objetivo general del proyecto terminal fue: *"Realizar una aproximación de la técnica de repixelización de arte pixelado, como la reportada por Kopf, J. y Lischinski, D.* ***[1]****, a través de la vectorización[4] y el procesamiento digital de imágenes."*

El objetivo general del proyecto terminal se cumplió; se logró procesar una imagen y obtener una notable mejora; mediante un cálculo basado tanto en la distancia entre los píxeles como en sus colores.

[2] Interpolación: Proceso de obtención de nuevos puntos partiendo del conocimiento de un conjunto discreto de puntos.
[3] Segmentación: Proceso de dividir una imagen digital en varias partes (grupos de píxeles) u objetos.
[4] Vectorización: Proceso de convertir una imagen formada por píxeles en una imagen formada por vectores.

Objetivos Específicos

- Convertir imágenes *Pixel Art*[5] en una representación de vectores independientes de la resolución.
- Calcular de forma automática la separación del fondo de la imagen en presencia de múltiples colores.
- Localizar y representar de forma precisa las curvas de nivel 1[6] que componen la imagen.
- Resolver correctamente las ambigüedades locales, al garantizar una curvatura geométricamente continua.

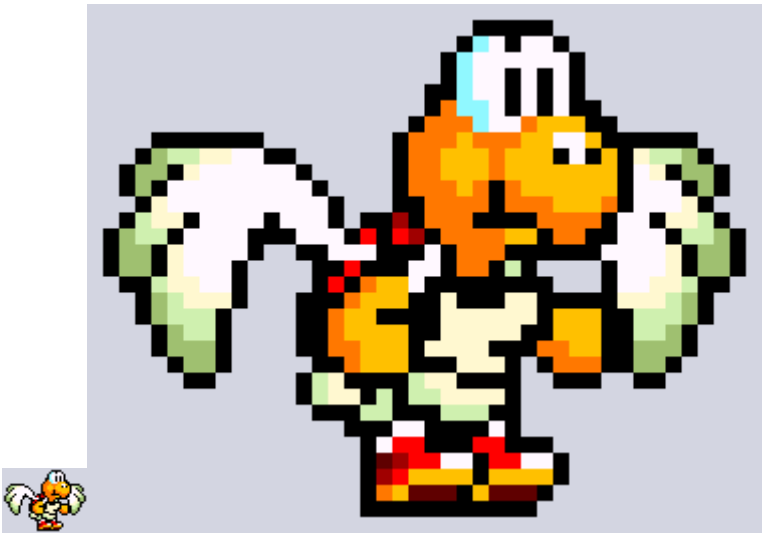
Finalmente, integrar los objetivos específicos en un software que funcione como aplicación de escritorio.

[5] *Pixel Art*: Expresión de arte de manera digital, donde los detalles de las imágenes son cuidadosamente representados al nivel de un píxel.
[6] Curvas de nivel 1 (del inglés *Level Sets*): Forma de describir bordes.

Algoritmo

El algoritmo que se desarrolló en el presente proyecto terminal se puede describir en los siguientes pasos:

1. Se toma la imagen de origen y se amplía (sin modificación alguna) según lo haya establecido el usuario. Para el presente ejemplo tomaremos el valor de "8x", esto quiere decir que la imagen resultante será 8 veces mayor que la original.



Imágenes 2 y 3. La imagen inicial es de 42x33 píxeles, mientras que la imagen ampliada es de 336x264 píxeles.

2. Ahora, en base a la nueva imagen, tomaremos el primer píxel de cada cuadro de 8x8 píxeles, tal como podemos ver en la imagen, el resto de los píxeles serán establecidos en blanco.

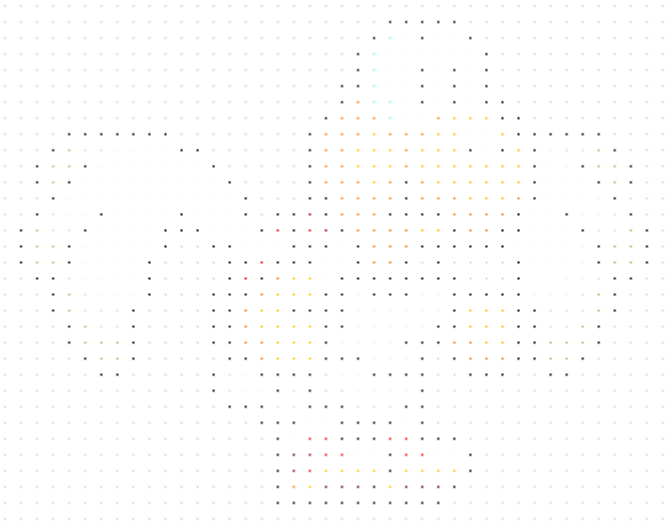


Imagen 4. Formada solamente con el primer píxel de cada cuadro de 8x8 píxeles.

3. Para crear el diagrama que se usará para obtener la imagen final es necesario recorrer la nueva imagen a lo ancho y a lo alto en pasos de 8 píxeles, en cada iteración buscaremos si alguno de los 8 vecinos del píxel en cuestión es del mismo color.

   En caso de que alguno de sus vecinos tenga color igual, trazaremos una línea que una a dichos vecinos; de esta manera obtendremos un diagrama como el que a continuación se muestra.

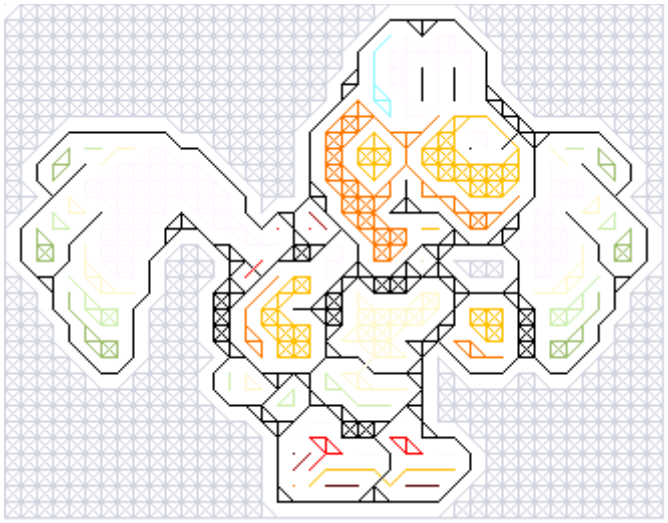


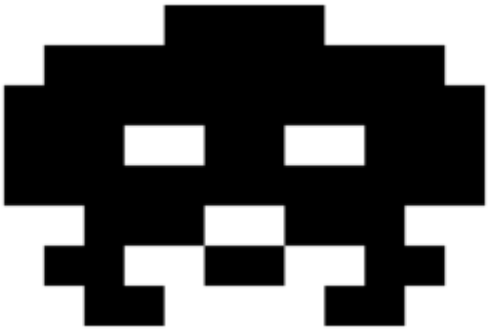
Imagen 5. Diagrama que usaremos para obtener la imagen final repixelada.

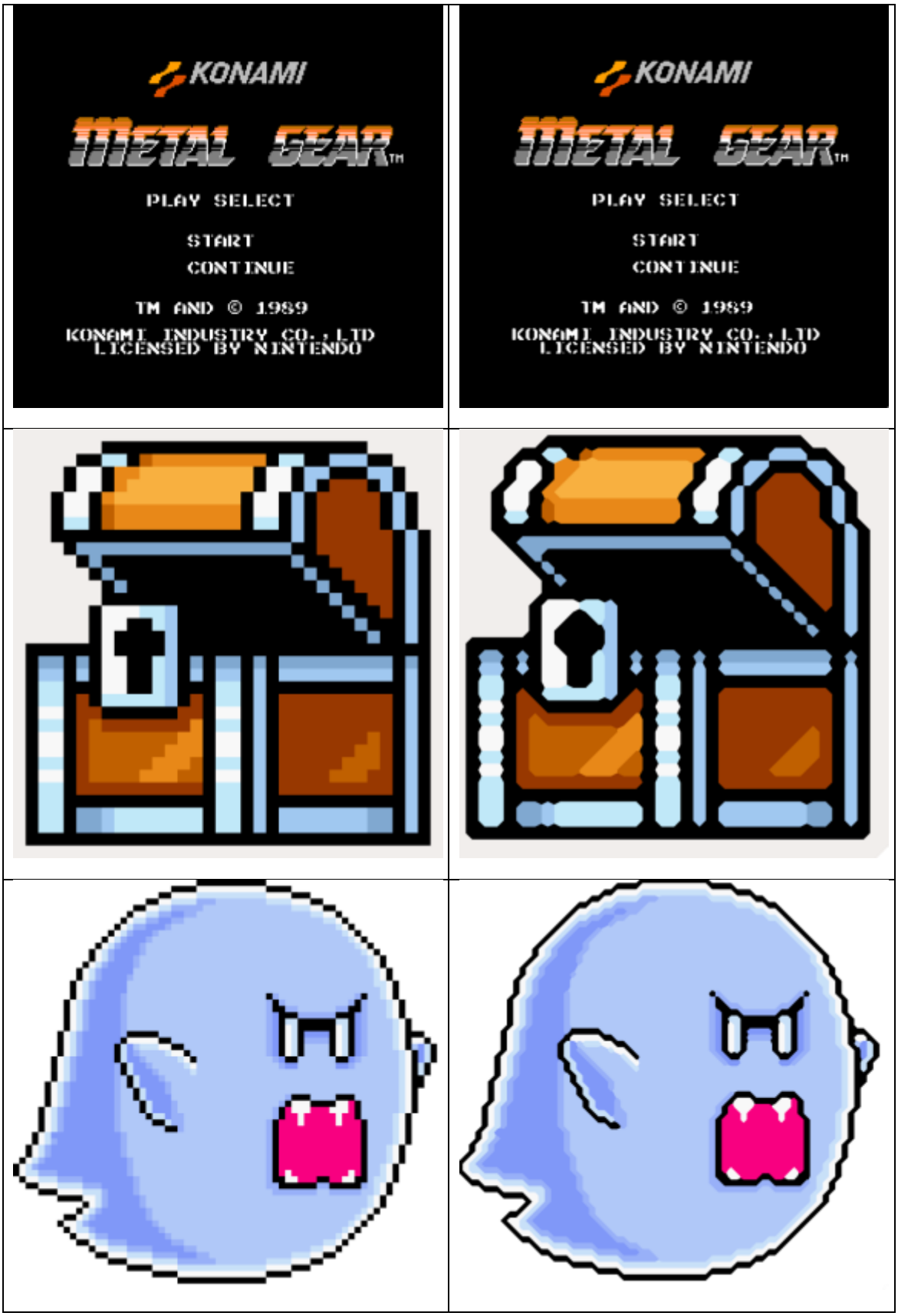
4. Una vez obtenido el diagrama, aplicaremos el algoritmo de Lloyd al diagrama, el cual nos generará la imagen final repixelada.

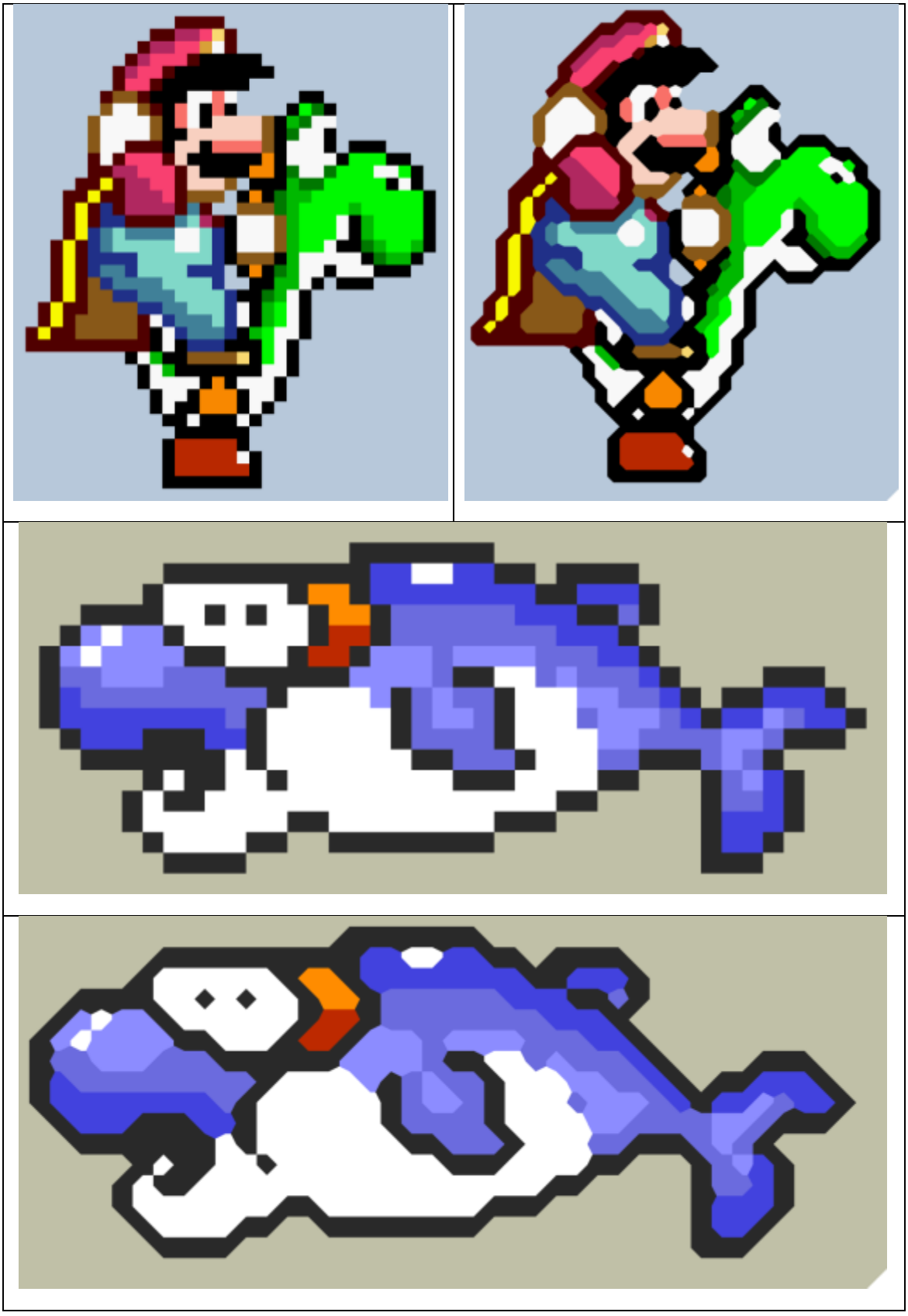
Imagen 6. Imagen final repixelada.

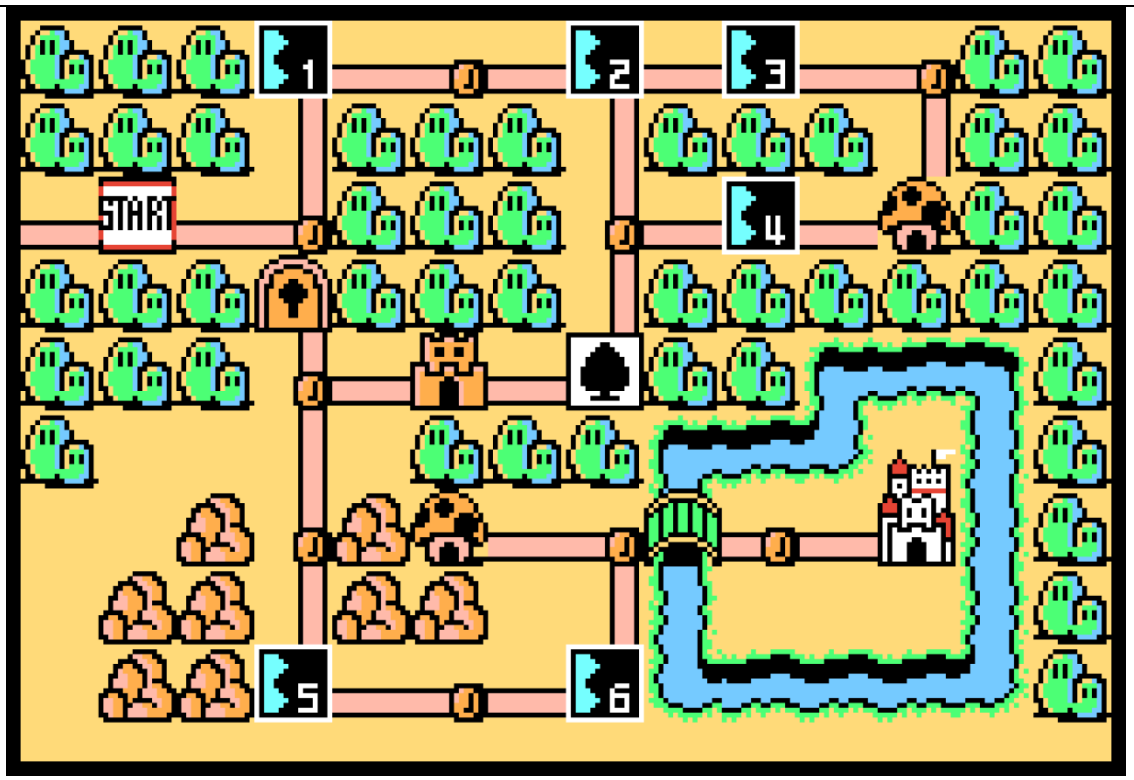
Resultados

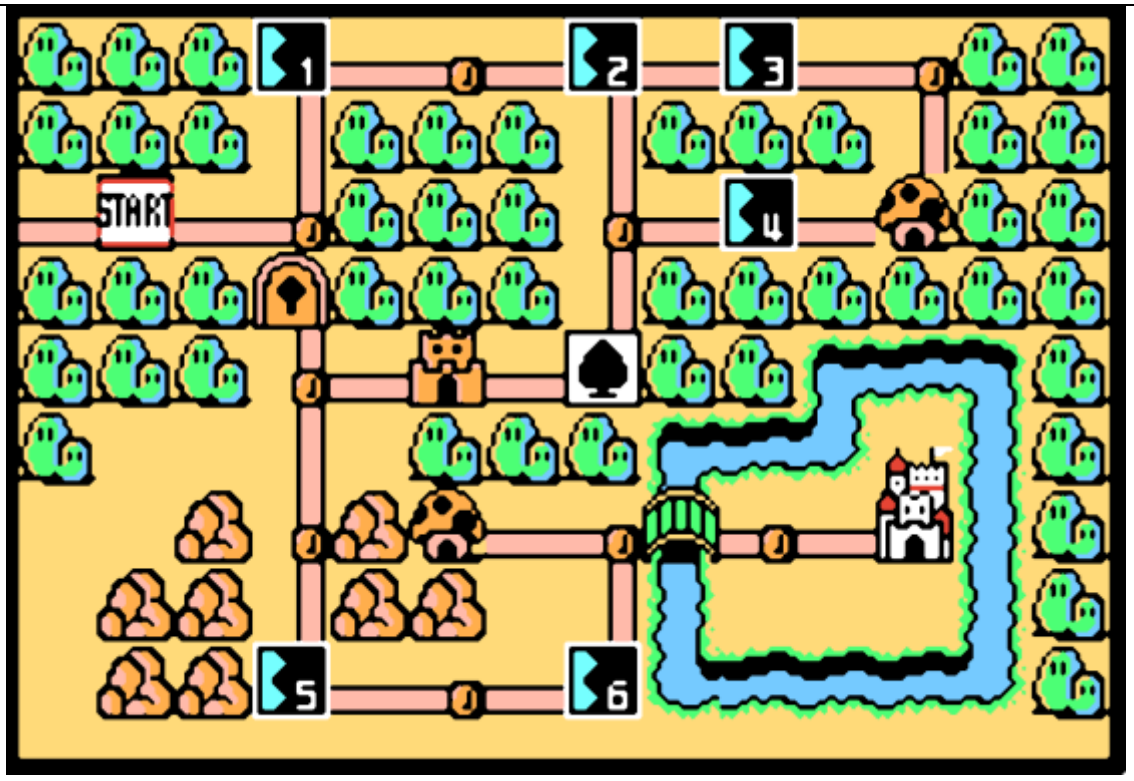
Imágenes Iniciales	Imágenes Finales
	
	



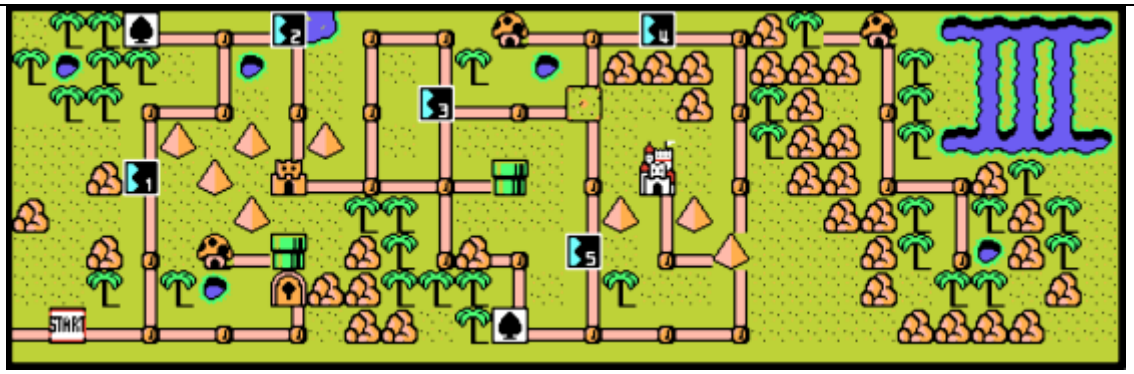

KONAMI
METAL GEAR
PLAY SELECT
START
CONTINUE
TM AND © 1989
KONAMI INDUSTRY CO., LTD
LICENSED BY NINTENDO
KONAMI
METAL GEAR
PLAY SELECT
START
CONTINUE
TM AND © 1989
KONAMI INDUSTRY CO., LTD
LICENSED BY NINTENDO

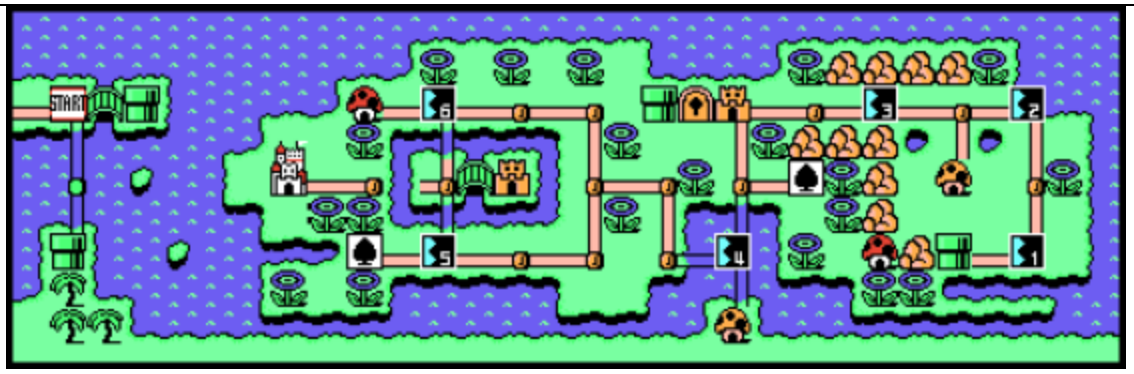


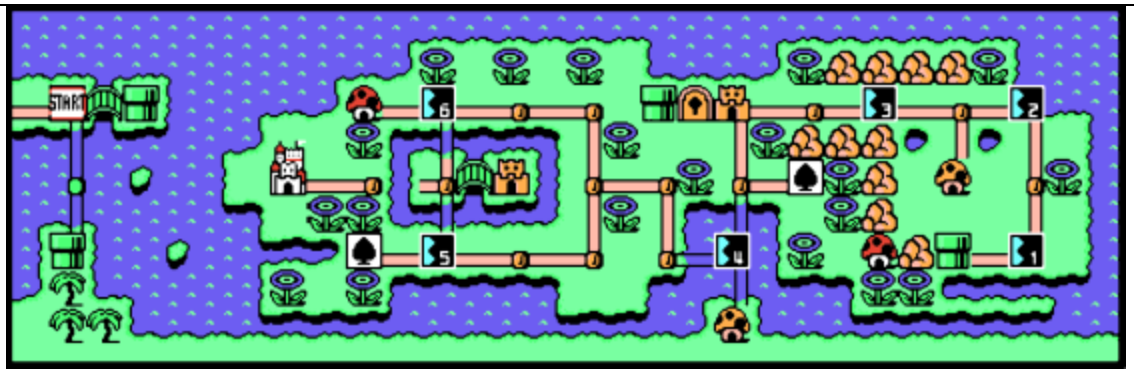

1
2
3
START
4
5
6

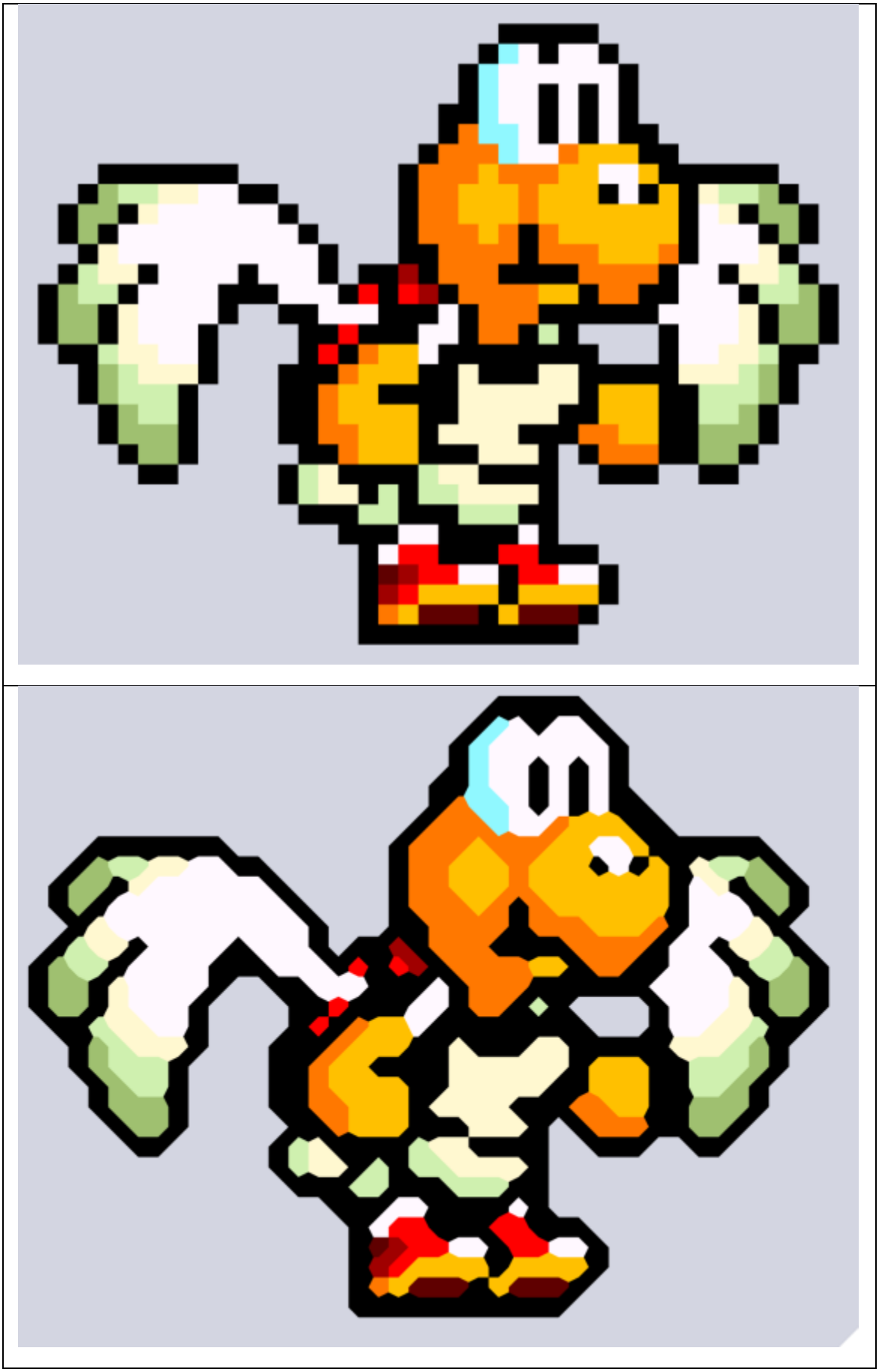

1
2
3
START
4
5
6


START


START


START


START



Imágenes que ilustran los resultados obtenidos con el presente proyecto.

Las imágenes mostradas anteriormente son resultado de la técnica que se diseñó e implemento durante el presente proyecto, a continuación se describe detalladamente dicha técnica.

Módulos de la Aplicación

El proyecto está compuesto por 3 módulos (Diagrama 1): el módulo generador de bordes, el módulo procesador y el módulo generador de imagen.

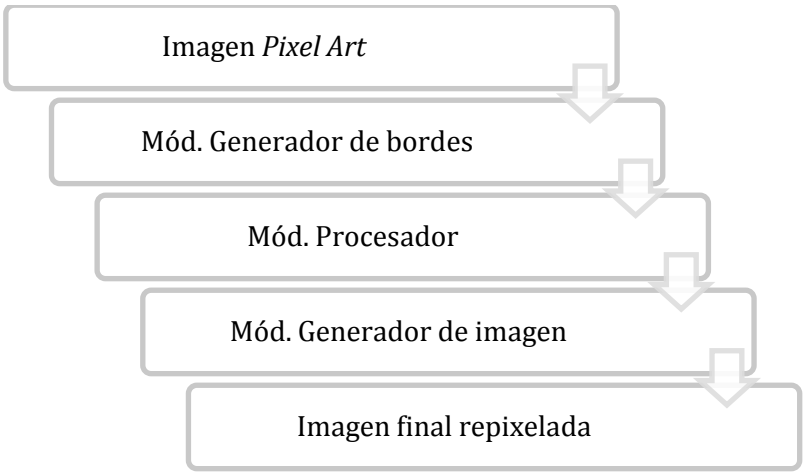


Diagrama 1. Módulos de la aplicación.

Módulo 1 – Generador de bordes.

Este módulo tiene 2 objetivos principales:

- Realizar el cálculo automático de la separación del fondo de la imagen.
- Detectar los bordes de la imagen y obtener las curvas de nivel 1 apropiadas.

Este módulo tiene como entrada una imagen *Pixel Art* en formato PNG y genera como salida la localización y estructura de los bordes que componen la imagen.

A continuación se ilustra la entrada y salida del módulo 1 (Imágenes 1 y 2).

Imagen 7. *Pixel Art* de entrada.

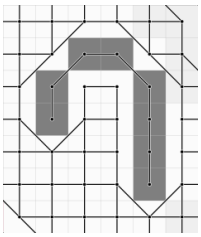


Imagen 8. Localización y estructura de los bordes.

Módulo 2 – Procesador.

Este módulo tiene 2 objetivos principales:

- Conectar los bordes vectorizados generados por el módulo anterior.
- Resolver correctamente las ambigüedades locales.

Este módulo tiene como entrada la localización y estructura de los bordes que componen la imagen y genera como salida la conexión de los bordes vectorizados.

A continuación se ilustra la entrada y salida del módulo 2 (Imágenes 3 y 4).

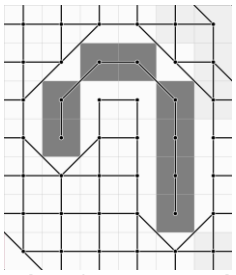


Imagen 9. Localización y estructura de los bordes.

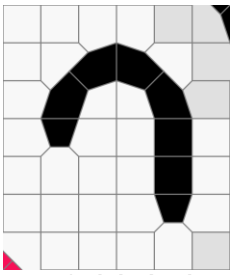


Imagen 10. Conexión de los bordes vectorizados.

Módulo 3 – Generador de imagen.

Este módulo tiene como objetivo almacenar el mapa de bits generado por el módulo anterior.

Este módulo tiene como entrada la conexión de los bordes vectorizados y genera como salida la imagen repixelada en formato PNG.

A continuación se ilustra la entrada y salida del módulo 3 (Imágenes 5 y 6).

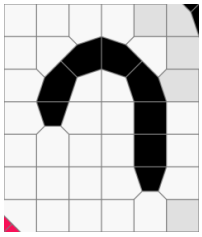


Imagen 11. Conexión de los bordes vectorizados.

Imagen 12. Imagen final repixelada.

Conclusiones

El presente proyecto presenta una implementación para obtener una repixelización de una imagen PNG[7] para así obtener una representación independiente de la resolución.

La principal razón por la que decidí realizar el presente proyecto tiene que ver con cómo el amplio crecimiento de cálculo computacional permite mejorar la experiencia del usuario en muchos aspectos, uno de ellos, los videojuegos.

Al realizar las primeras pruebas con archivos PNG surgieron algunos imprevistos, sin embargo, los problemas antes mencionados lograron resolverse y el desarrollo de la aplicación tuvo un resultado satisfactorio.

La aplicación desarrollada durante este proyecto puede ser mejorada al implementar un algoritmo que sea capaz de evaluar correctamente regiones de 2x2 píxeles cuyas diagonales sean de colores diferentes; lo cual daría a la imagen una aproximación aún mayor. Las siguientes imágenes ilustran dicho problema:

Imágenes que ilustran el problema de las ambigüedades locales.

El problema radica en saber qué línea pertenece a una región mayor. En la imagen izquierda: ¿Debe conservarse la línea roja o la blanca?; tenemos el mismo problema con la imagen derecha.

Finalmente, es importante señalar que este proyecto logró cumplir con los objetivos generales y particulares del proyecto, pues es capaz de procesar una imagen y obtener una notable mejora; a través de la vectorización y el procesamiento digital de imágenes.

Para ilustrar las capacidades del proyecto de una forma más clara se incluye un video a 60fps; del lado izquierdo se muestran imágenes sin modificación y del derecho imágenes repixeladas.

[7] PNG (del inglés *Portable Network Graphics*): Formato de imagen que utiliza compresión sin pérdida.

Bibliografía

[1] Kopf, J. & Lischinski, D. "Depixelizing Pixel Art". *ACM Trans. Graph.* vol. 30, no. 4, article 99, Julio 2011.

[2] Liauw Kie Fa, D. (2011.10.30) *2xSaI: The advanced 2x Scale and Interpolation*

Engine. [En línea]. Disponible: http://vdnoort.home.xs4all.nl/emulation/2xsai/

[3] Mazzoleni, A. (2011.10.30) *Scale2x*. [En línea].

Disponible: http://scale2x.sourceforge.net

[4] Stepin, M. (2011.10.30) *Demos & Docs – hq2x/hq3x/hq4x Magnification Filter*. [En línea]. Disponible: http://www.hiend3d.com/demos.html

[5] F. M. Díaz Cabrera, "Clasificador de objetos en banda infinita por medio de procesamiento digital de imágenes", Proyecto Terminal, División de CBI, Universidad Autónoma Metropolitana Azcapotzalco, México, 2008.

[6] M. A. Rojas Sandoval, "Algoritmo para acoplamiento automático por similitudes de imágenes digitales", Proyecto Terminal, División de CBI, Universidad Autónoma Metropolitana Azcapotzalco, México, 2010.

Apéndice A
Manual de Instalación

Requisitos del sistema

Arquitecturas admitidas

- x86
- x64

Hardware

- Procesador: 1 GHz o superior
- RAM: 512 MB de RAM o superior
- Resolución: 1366x768 o superior

Software

- Microsoft .NET Framework 4
- Windows Installer 3.1
- Windows XP SP3/Vista SP1/7/8

Instalación

Para instalar la aplicación es necesario:

- Microsoft .NET Framework 4
- Windows Installer 3.1

Para comenzar la instalación es necesario ejecutar alguno de los siguientes archivos:

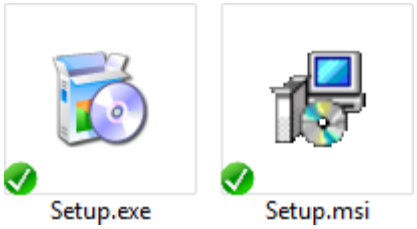


Imagen 1. Instaladores de la aplicación en formato EXE y MSI.

Si no están instalados los requisitos de software la aplicación asesorará al usuario para su instalación.

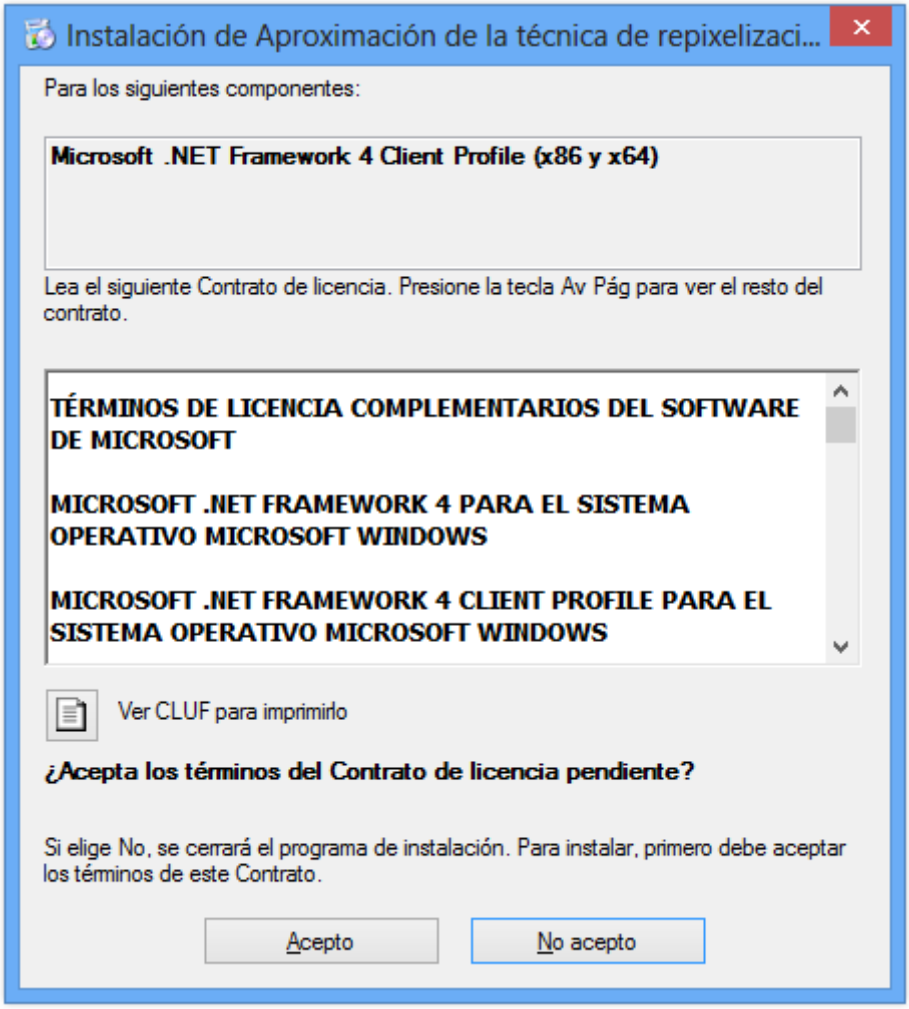


Imagen 2. El asistente verificará si los requisitos de software de cumplen.

Una vez que se tienen los requisitos instalados, el asistente automáticamente procederá con la instalación.

Se desplegará un asistente para instalación. Clic en “Siguiente >”.

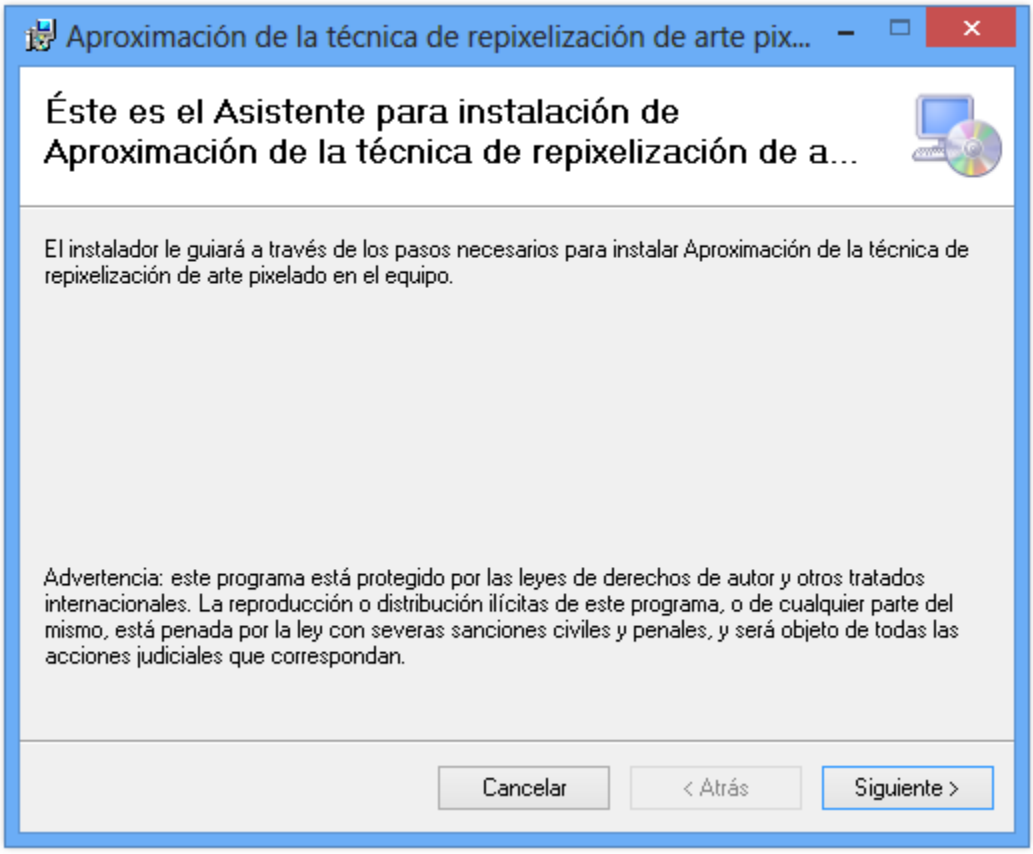


Imagen 3. Asistente para instalación.

A continuación clic en “Siguiente >” en caso de no querer cambiar la carpeta de instalación; en caso de querer instalarlo en alguna carpeta especifica seleccionarla en la opción: “Examinar...” y después clic en “Siguiente >”.

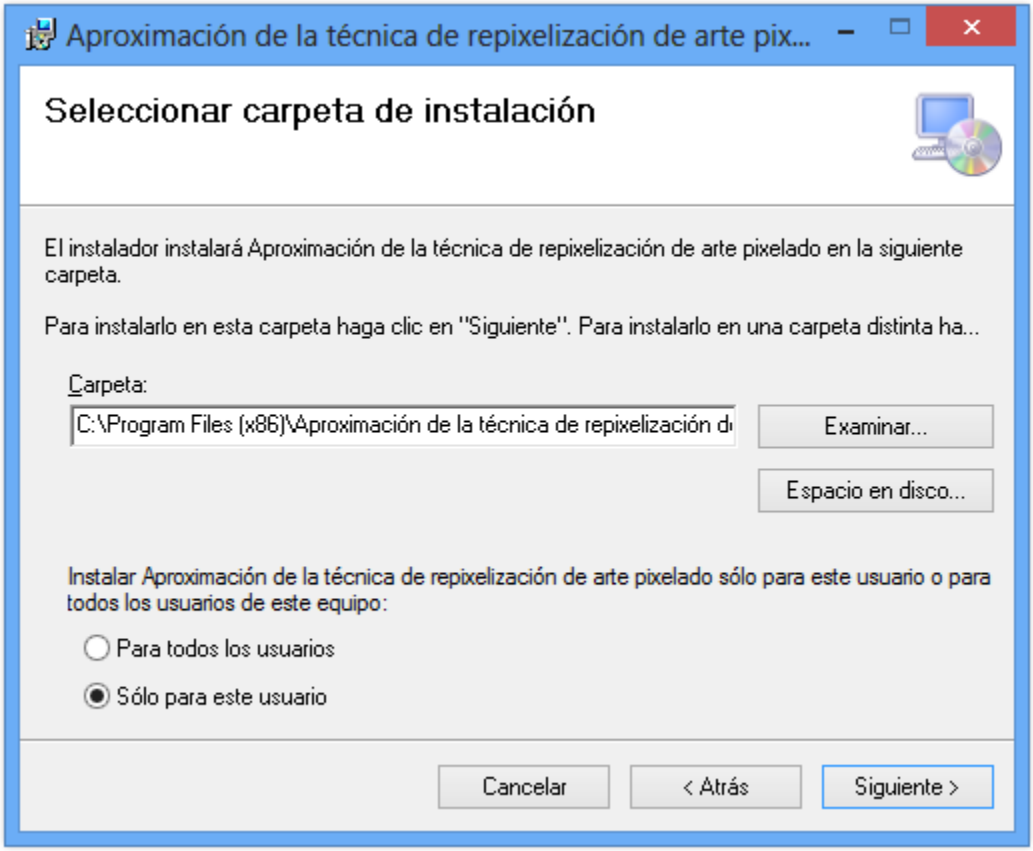


Imagen 4. Asistente para instalación.

Dependiendo de la arquitectura del sistema operativo, el programa se instalará en:

- x86
  C:\Program Files (x86)\Aproximación de la técnica de repixelización de arte pixelado
- x64 C:\Program Files\Aproximación de la técnica de repixelización de arte pixelado

Por ultimo confirmamos la instalación seleccionando “Siguiente >”.

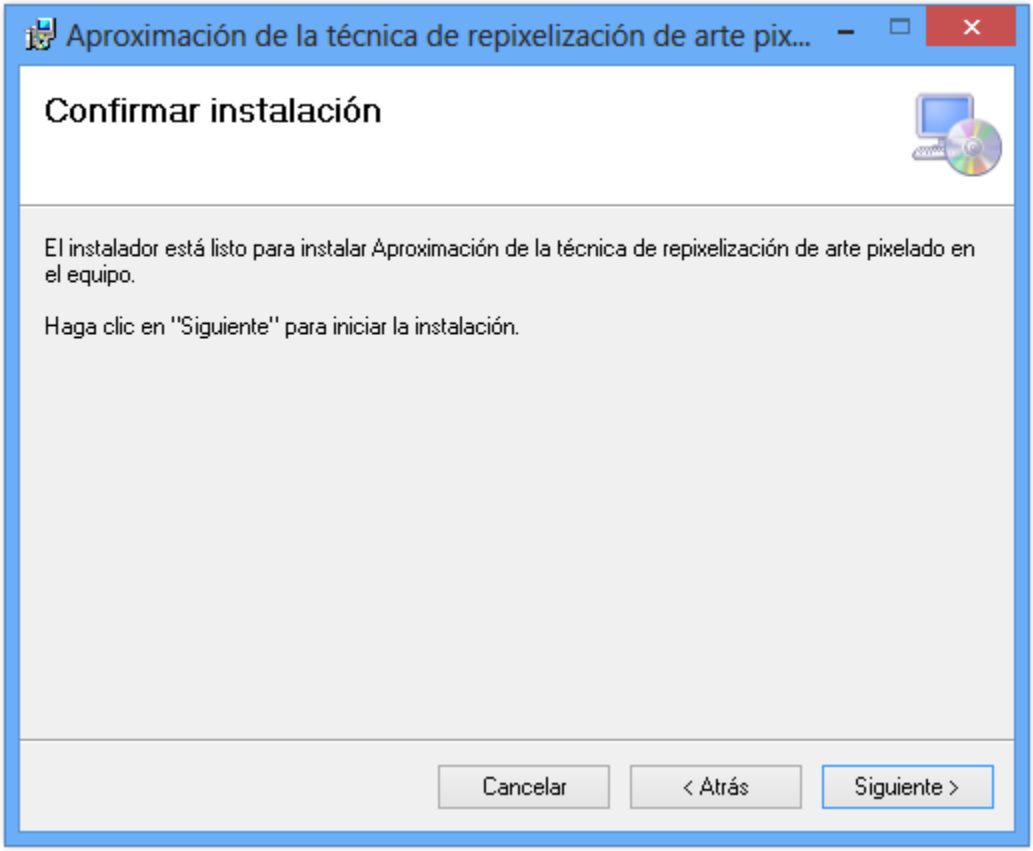


Imagen 5. Asistente para instalación.

Finalmente se muestra una pantalla confirmándonos que la instalación está completa.

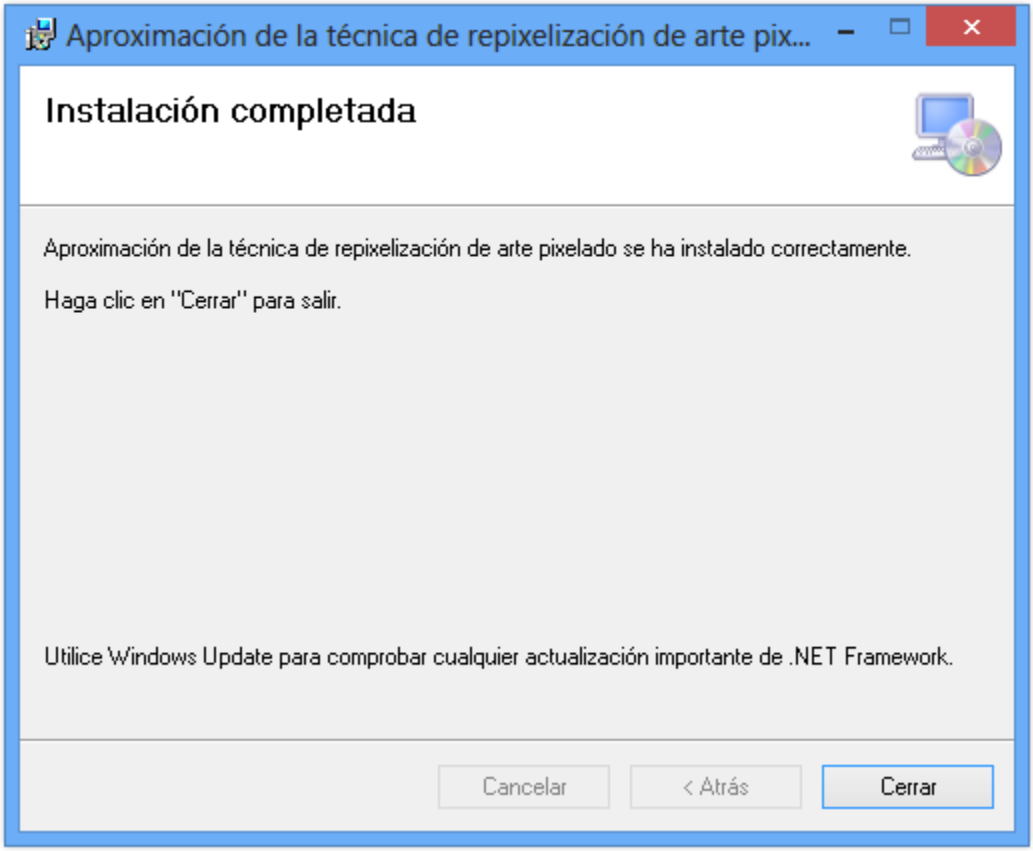


Imagen 6. Asistente para instalación.

Ejecutar la aplicación

Para abrir la aplicación tenemos 2 opciones:

- Abrir el acceso directo del escritorio.
- Menú Inicio: Todos los programas, Proyecto Terminal y seleccionamos "Aproximación de la técnica de repixelización de arte pixelado".

Apéndice B
Manual de Usuario

Introducción

El presente manual describe cómo utilizar la interfaz de la aplicación: *"Proyecto Terminal - Aproximación de la técnica de repixelización de arte pixelado"*.

Interfaz

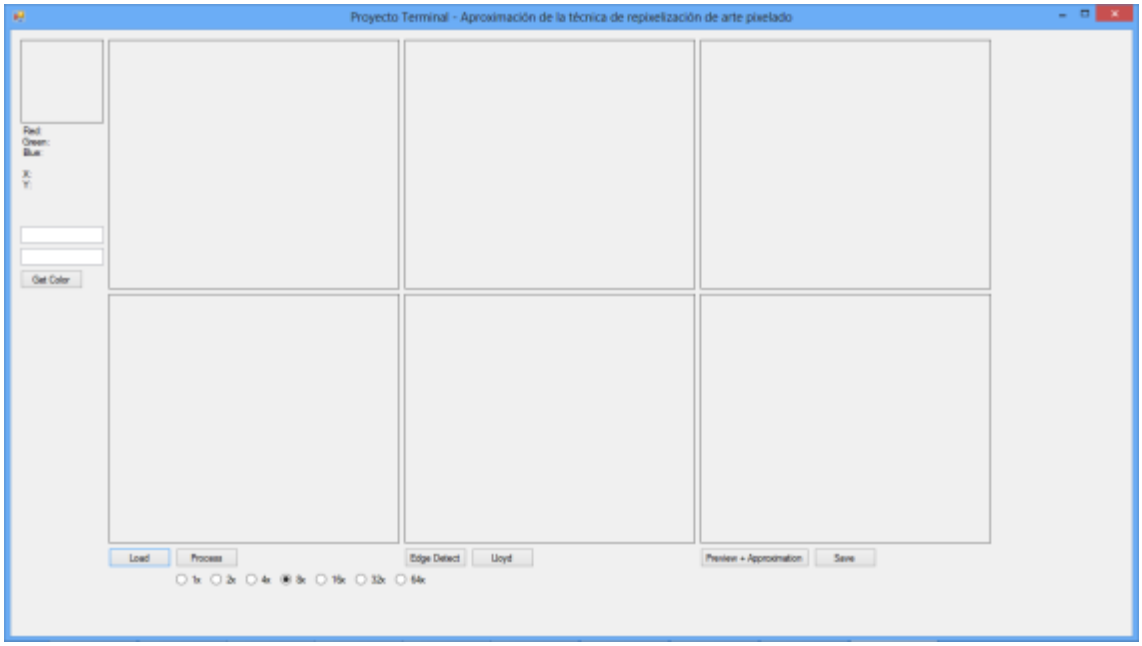


Imagen 1. Interfaz de la aplicación.

- Botón "Load": Permite elegir una imagen en formato PNG que se usará a lo largo del programa. Si se elige una imagen correcta:
    1. PictureBox1 mostrará una copia sin ninguna modificación.
    2. PictureBox2 mostrará una ampliación sin ninguna modificación ("x" veces, especificado en los parámetros).
    3. PictureBox3 mostrará una ampliación modificada ("x" veces, especificado en los parámetros).

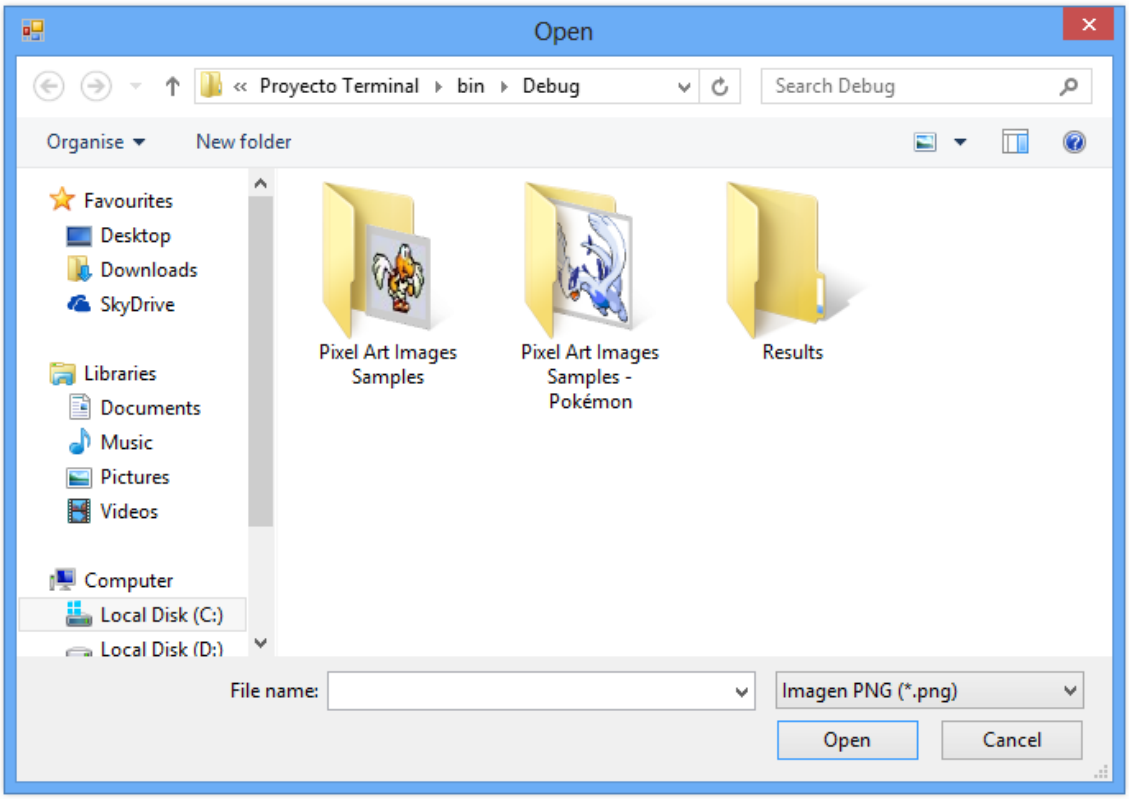


Imagen 2. Interfaz de la aplicación; ventana para elegir la imagen que se usará.

Si se elige algún archivo incorrecto, ya sea una imagen en formato diferente a PNG o algún archivo corrupto la aplicación mostrará un mensaje con el error correspondiente.

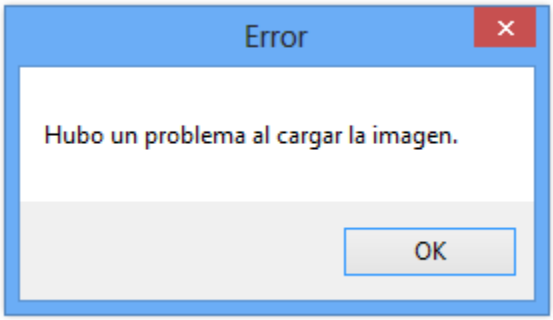


Imagen 3. Interfaz de la aplicación; ventana que muestra un error al cargar la imagen.

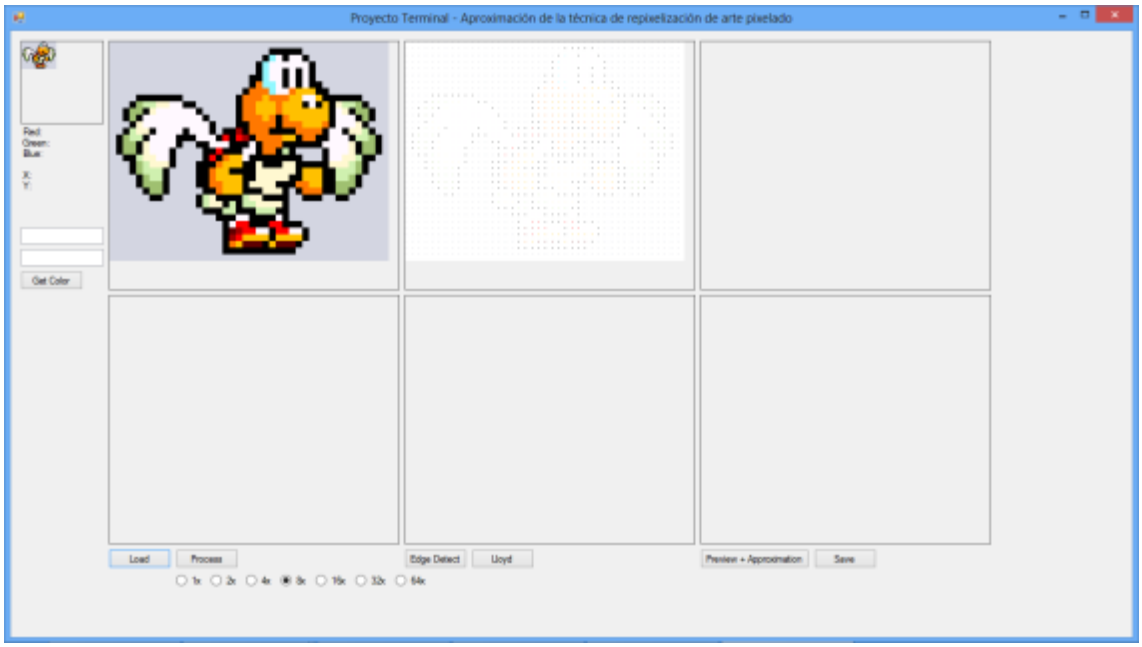


Imagen 4. Interfaz de la aplicación después de cargar exitosamente una imagen.

- Botón "Process": Permite actualizar el tamaño de la imagen que se usará a lo largo del programa ("x" veces, especificado en los parámetros):
    1. PictureBox2 mostrará la imagen sin modificar con el nuevo tamaño.
    2. PictureBox3 mostrará la imagen modificada con el nuevo tamaño.

- Botón "Lloyd": Permite calcular el algoritmo de Lloyd de la imagen:
    1. PictureBox4 mostrará la imagen resultante de aplicar el algoritmo de Lloyd a la imagen del PictureBox3.

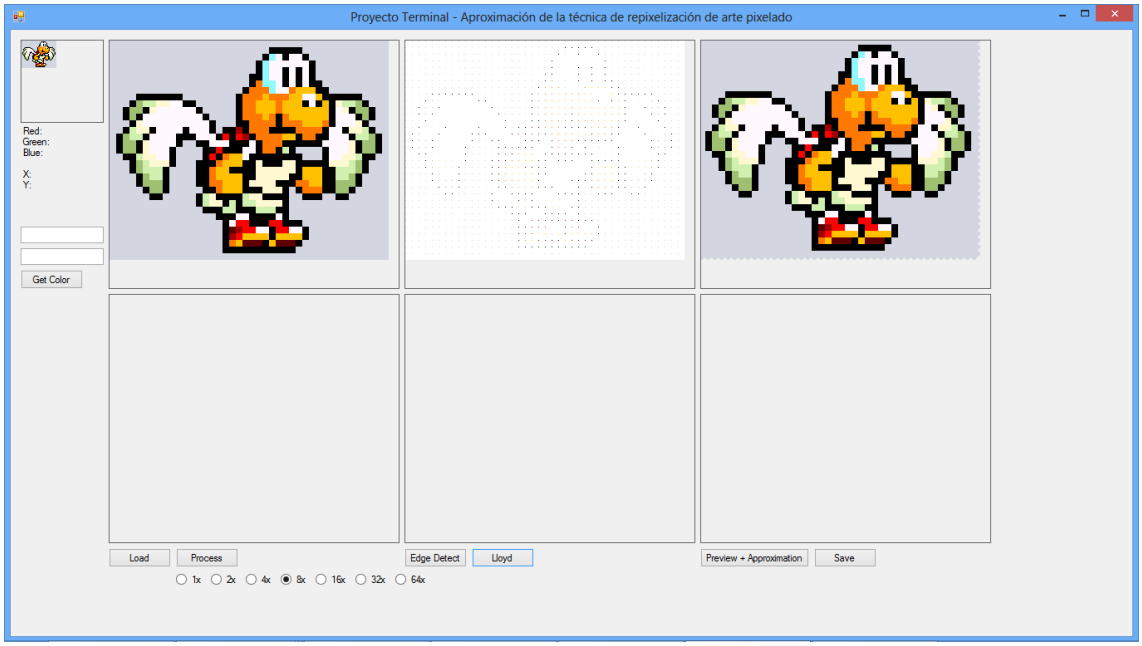


Imagen 5. Interfaz de la aplicación después de mostrar el cálculo del algoritmo de Lloyd.

- Botón "Edge Detect": Permite calcular el algoritmo de Canny de la imagen:
    1. PictureBox5 mostrará la imagen resultante de aplicar el algoritmo de Canny a la imagen del PictureBox3.

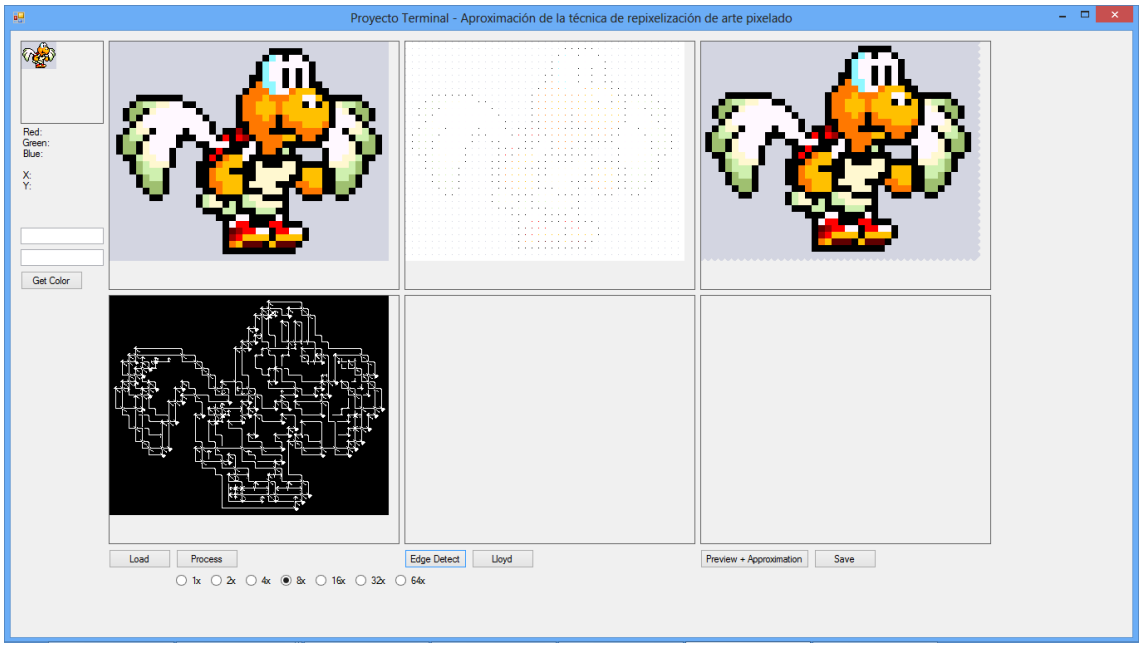


Imagen 6. Interfaz de la aplicación después de mostrar el cálculo del algoritmo de Canny.

- Botón "Preview+Approximation": Permite calcular la aproximación de la imagen que se quiere obtener:
    1. PictureBox6 mostrará el diagrama que se usará para obtener la imagen aproximada.
    2. Por último, PictureBox7 mostrará la imagen final procesada.

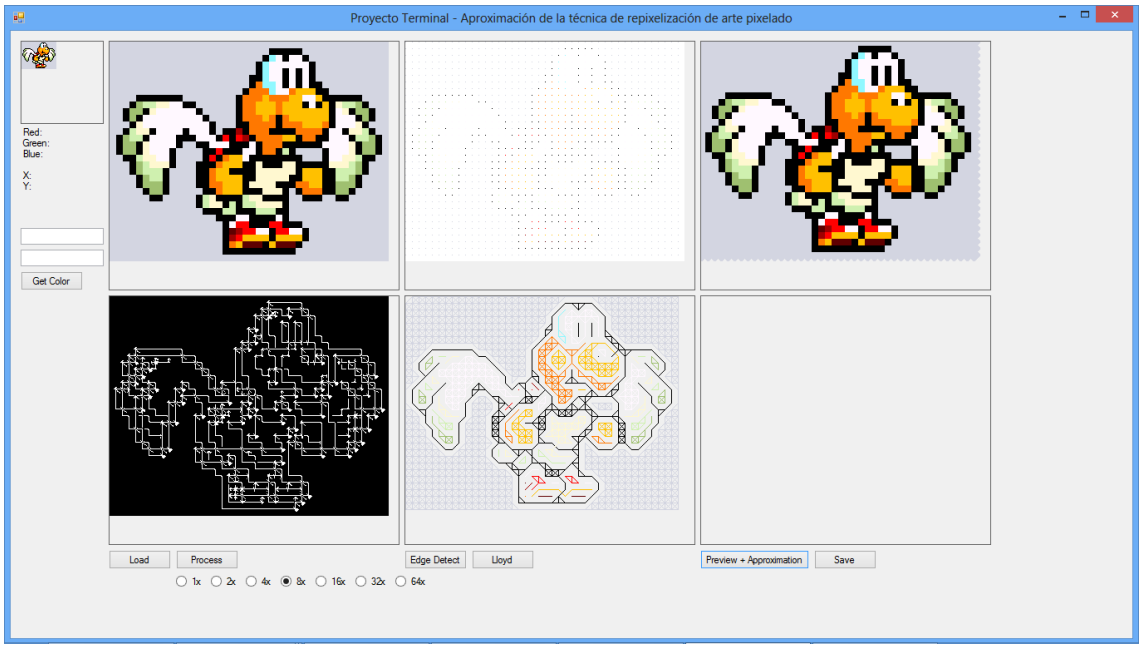


Imagen 7. Interfaz de la aplicación después de mostrar el cálculo del diagrama.

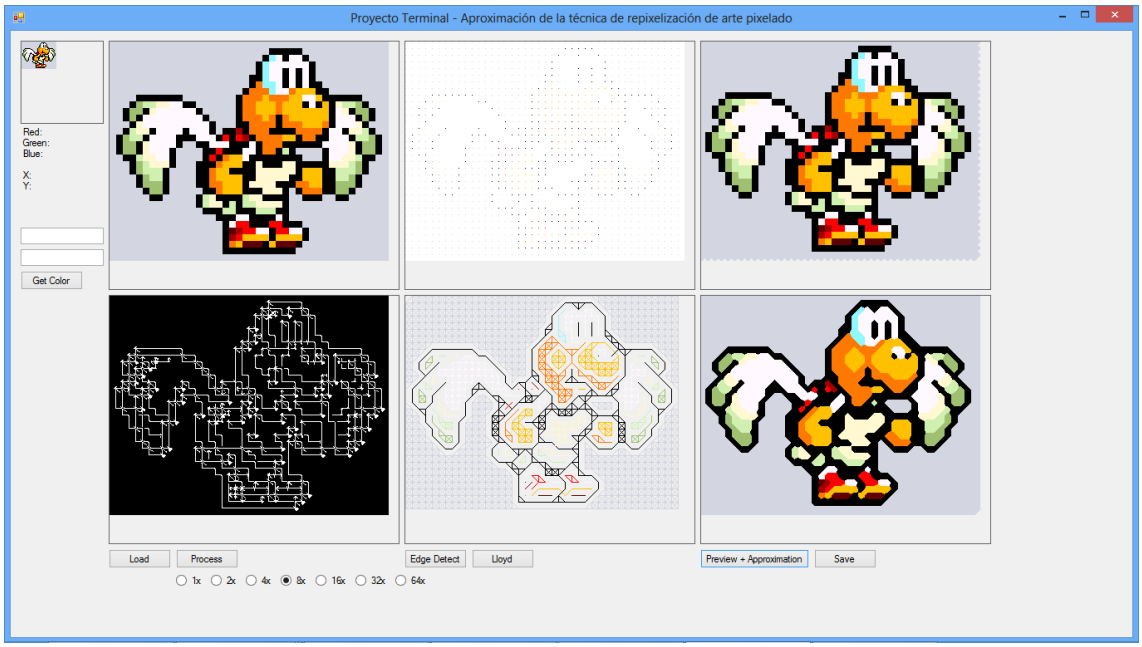


Imagen 8. Interfaz de la aplicación después de mostrar el cálculo de la imagen final.

- Botón "Save": Permite al usuario almacenar los resultados:
    1. Primero se le dará la opción de almacenar la imagen original ampliada "x" veces sin modificación alguna.
    2. Por último, se le dará la opción de almacenar la imagen final procesada.

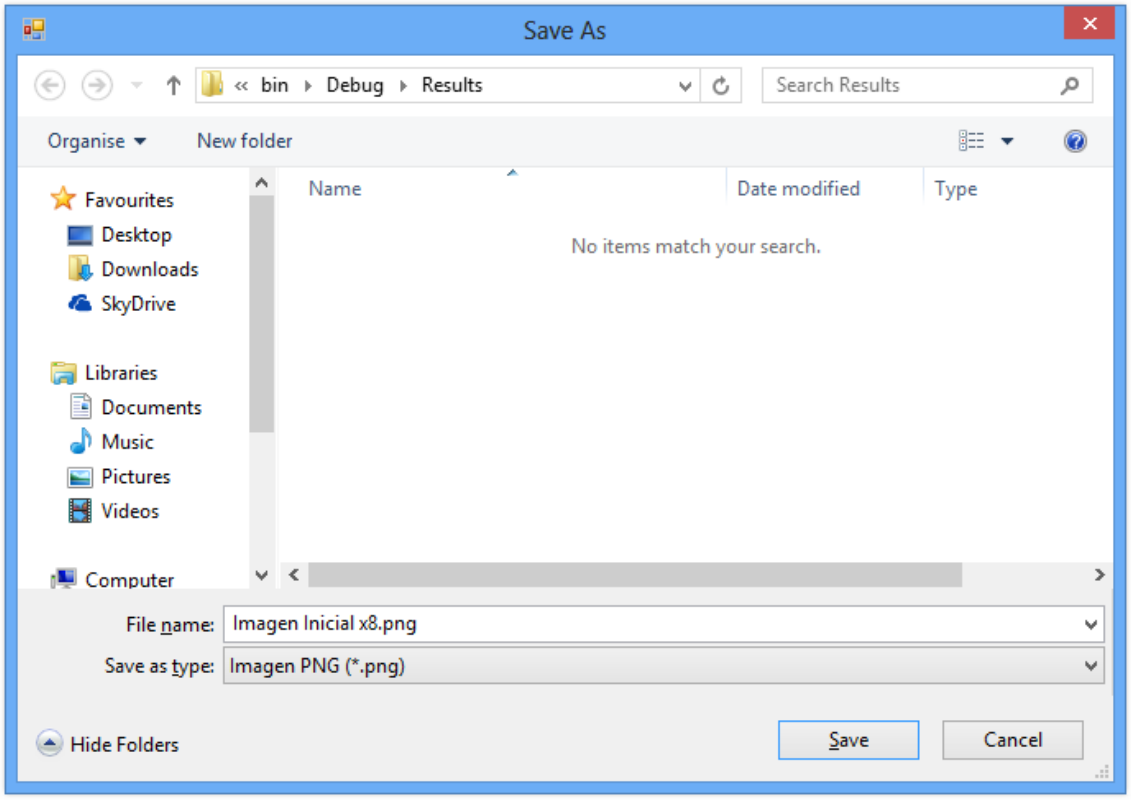


Imagen 9. Interfaz de la aplicación; ventana para elegir donde se guardará la imagen original.

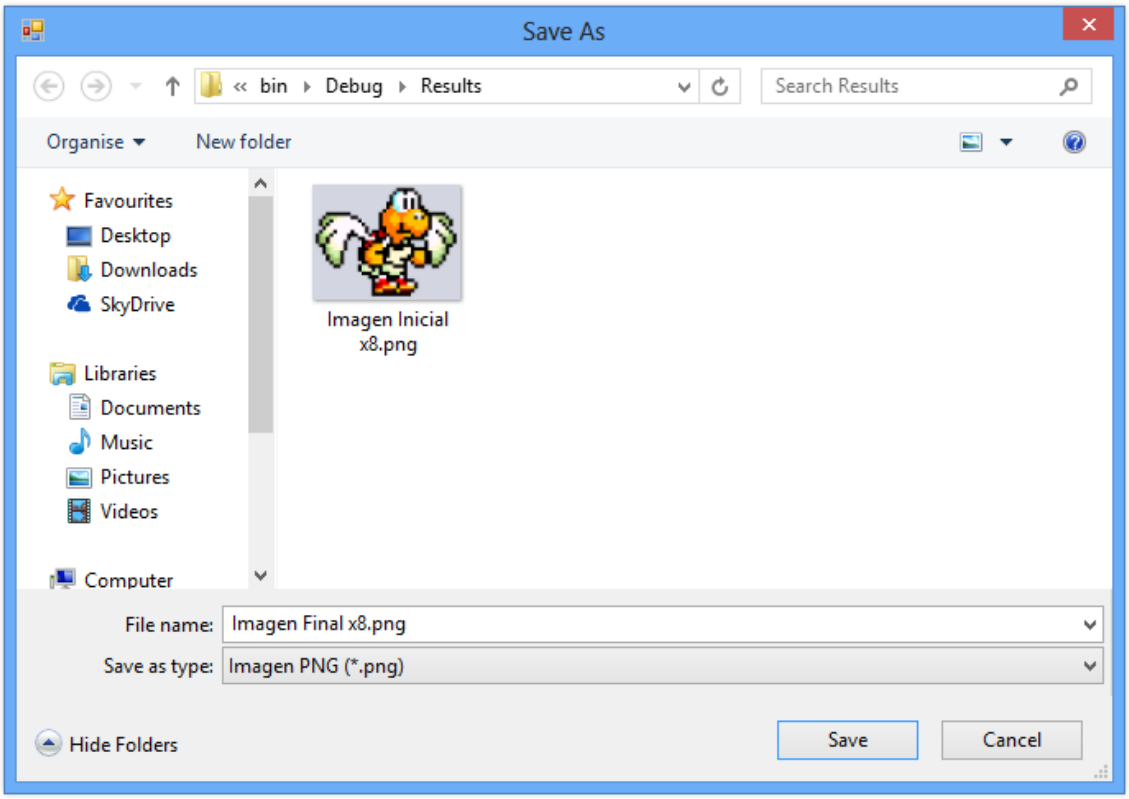


Imagen 10. Interfaz de la aplicación; ventana para elegir donde se guardará la imagen final.

- Botón "GetColor": Permite al usuario obtener colores de los píxeles, los cuales se mostraran debajo del PictureBox1:
    1. Usando los cuadros de texto, el usuario puede establecer coordenadas para obtener los componentes del color de dicho píxel; también puede hacer un clic en el PictureBox1 obteniendo los mismos resultados.

Imagen 11. Interfaz de la aplicación; ventana con información de los componentes RGB de la imagen.

Apéndice C
Manual del Programador

Introducción

El presente manual describe de forma rápida y sencilla los módulos, clases y métodos que se usaron durante el desarrollo de la aplicación: *"Proyecto Terminal - Aproximación de la técnica de repixelización de arte pixelado"*.

Módulos de la Aplicación

El proyecto está compuesto por 3 módulos (Diagrama 1): el módulo generador de bordes, el módulo procesador y el módulo generador de imagen.

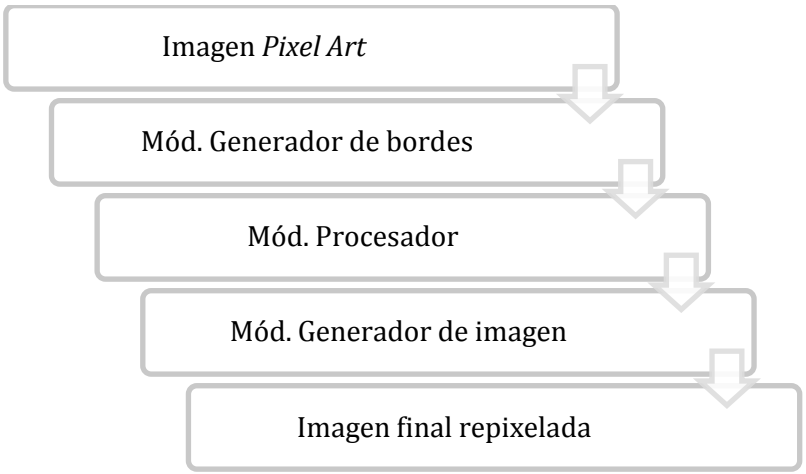


Diagrama 1. Módulos de la aplicación.

Módulo 1 – Generador de bordes.

Este módulo tiene 2 objetivos principales:

- Realizar el cálculo automático de la separación del fondo de la imagen.
- Detectar los bordes de la imagen y obtener las curvas de nivel 1 apropiadas.

Este módulo tiene como entrada una imagen *Pixel Art* en formato PNG y genera como salida la localización y estructura de los bordes que componen la imagen.

A continuación se ilustra la entrada y salida del módulo 1 (Imágenes 1 y 2).

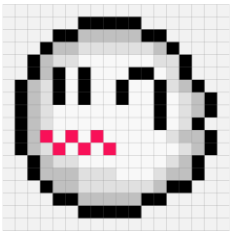


Imagen 1. *Pixel Art* de entrada.

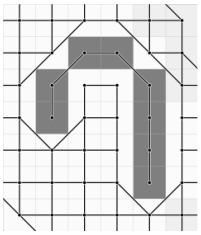


Imagen 2. Localización y estructura de los bordes.

Módulo 2 – Procesador.

Este módulo tiene 2 objetivos principales:

- Conectar los bordes vectorizados generados por el módulo anterior.
- Resolver correctamente las ambigüedades locales.

Este módulo tiene como entrada la localización y estructura de los bordes que componen la imagen y genera como salida la conexión de los bordes vectorizados.

A continuación se ilustra la entrada y salida del módulo 2 (Imágenes 3 y 4).

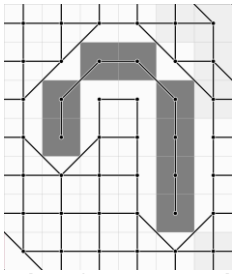


Imagen 3. Localización y estructura de los bordes.

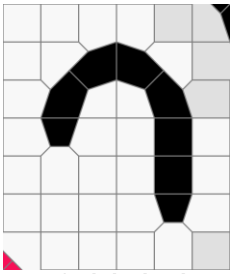


Imagen 4. Conexión de los bordes vectorizados.

Módulo 3 – Generador de imagen.

Este módulo tiene como objetivo almacenar el mapa de bits generado por el módulo anterior.

Este módulo tiene como entrada la conexión de los bordes vectorizados y genera como salida la imagen repixelada en formato PNG.

A continuación se ilustra la entrada y salida del módulo 3 (Imágenes 5 y 6).

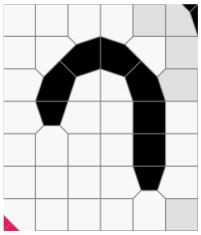


Imagen 5. Conexión de los bordes vectorizados.

Imagen 6. Imagen final repixelada.

Clase PT

Esta es la clase principal, tanto de la aplicación como del formulario; contiene todos los módulos y métodos de la aplicación.

Variables			
Nombre	Tipo	Nivel de accesibilidad	Descripción
imageTest	Bitmap	private	Verificar que la imagen original sea una verdadera imagen PNG.
imageLoaded	Bitmap	private	Imagen inicial sin modificación.
imagePictureBox2	Bitmap	private	Imagen ampliada sin modificación.
imagePictureBox3	Bitmap	private	Imagen ampliada modificada.
imagePictureBox4	Bitmap	private	Imagen que mostrará el resultado del algoritmo de Lloyd de la imagen.
imagePictureBox5	Bitmap	private	Imagen que mostrará el resultado del algoritmo de Canny de la imagen.
imagePictureBox6	Bitmap	private	Diagrama que se usará para obtener la imagen aproximada.
imagePictureBox7	Bitmap	private	Imagen final repixelada.
white	Color	private	Color blanco.
pixelOrigen pixelDestino	Color	private	Color del píxel de la coordenada x. Color del píxel de la coordenada y.
g	Graphics	private	Encapsula una superficie de dibujo.
pen	Pen	private	Define un objeto utilizado para dibujar líneas y curvas.
point1 point2	Point	private	Representa un par ordenado de coordenadas enteras x e y.

Tabla 1. Lista de variables usadas durante el desarrollo de la aplicación.

Métodos			
Nombre	Return	Nivel de accesibilidad	Descripción
loadButton_Click(object sender, EventArgs e)	void	private	Módulo 1. Generador de bordes. Este método realiza la carga de la imagen y localiza la estructura de los bordes.
processButton_Click(object sender, EventArgs e)	void	private	Este método permite reajustar la proporción que se usará sin tener que cargar nuevamente la imagen.
edgeDetectButton_Click(object sender, EventArgs e)	void	private	Este método realiza el cálculo del algoritmo de Canny de la imagen que se cargó.
lloydButton_Click(object sender, EventArgs e)	void	private	Este método realiza el cálculo del algoritmo de Lloyd de la imagen que se cargó.
previewButton_Click(object sender, EventArgs e)	void	private	Módulo 2. Procesador. Este método conecta los bordes calculados por el módulo 1 y calcula la imagen final repixelada.
saveButton_Click(object sender, EventArgs e)	void	private	Módulo 3. Generador de imagen. Este método genera como salida la imagen repixelada en formato PNG.
getColorButton_Click(object sender, EventArgs e)	void	private	Este método permite obtener los componentes de color de las coordenadas de la imagen que se especifiquen en los textBox.
pictureBox1_Click(object sender, EventArgs e)	void	private	Este método permite hacer clic en cualquier parte de la imagen y obtener los componentes de color de dichas coordenadas de la imagen.

Tabla 2. Lista de métodos usados durante el desarrollo de la aplicación.

Apéndice D
Guía Rápida del Producto

Introducción

La presente guía describe de forma rápida y sencilla cómo utilizar la aplicación: *"Proyecto Terminal - Aproximación de la técnica de repixelización de arte pixelado"*.

Interfaz

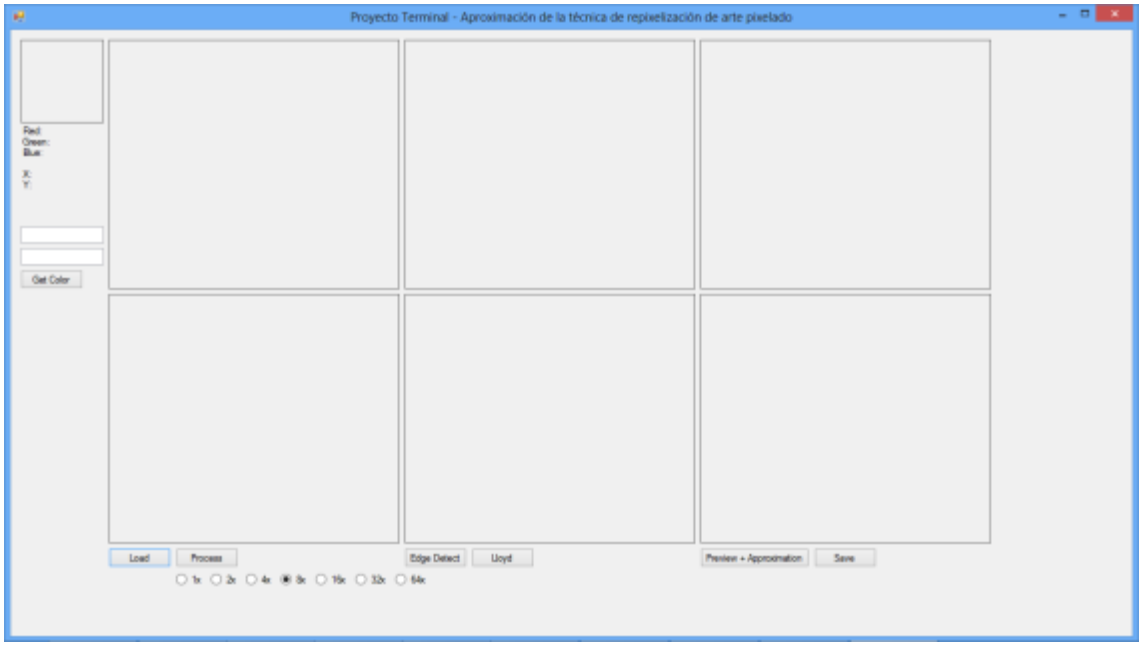


Imagen 1. Interfaz de la aplicación.

Lista de elementos que se muestran en la aplicación:

1. Área de selección de la proporción que se usará a lo largo del programa.
2. Botón para cargar una imagen.
3. Botón para procesar la imagen seleccionada.
4. Botón para calcular el algoritmo de Canny de la imagen seleccionada.
5. Botón para calcular el algoritmo de Lloyd de la imagen seleccionada.
6. Botón para calcular la imagen repixelada.
7. Botón para guardar la imagen original y la imagen repixelada.
8. Botón para calcular los componentes de color de un píxel de la imagen seleccionada.

Primero elegiremos la proporción que se usará a lo largo del programa; después cargaremos una imagen PNG usando el botón “Load”.

Una vez cargada correctamente la imagen PNG seleccionada, hacemos clic en el botón “Preview + Application”.

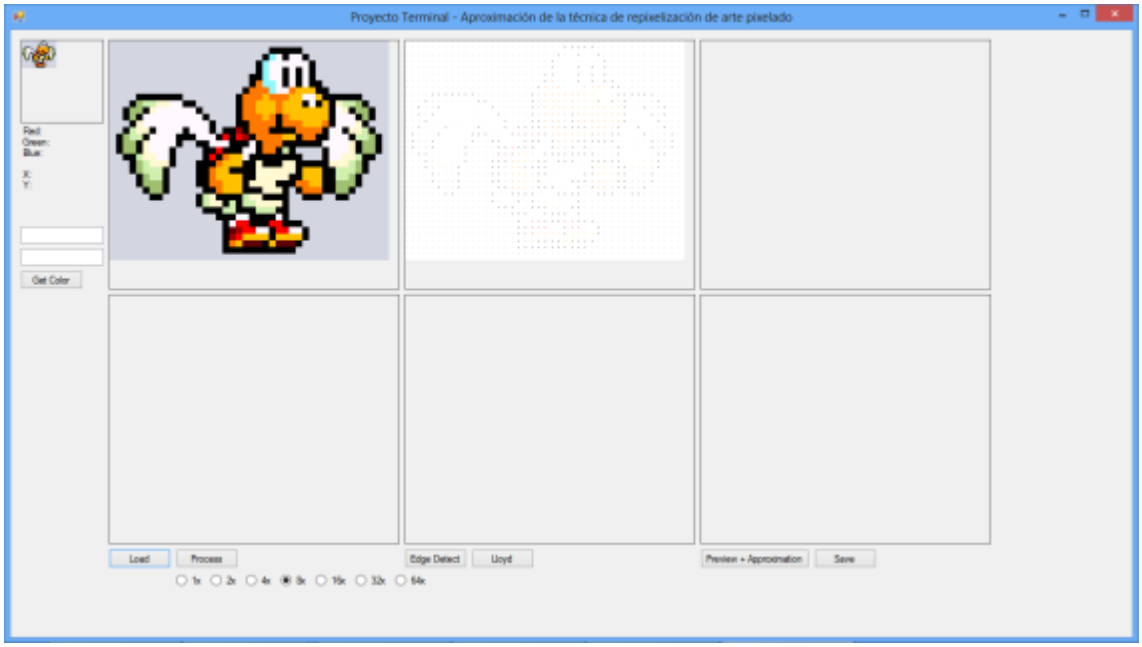


Imagen 2. Interfaz de la aplicación después de cargar exitosamente una imagen.

La aplicación mostrará en el Área 6 el diagrama que se usará para obtener la imagen repixelada.

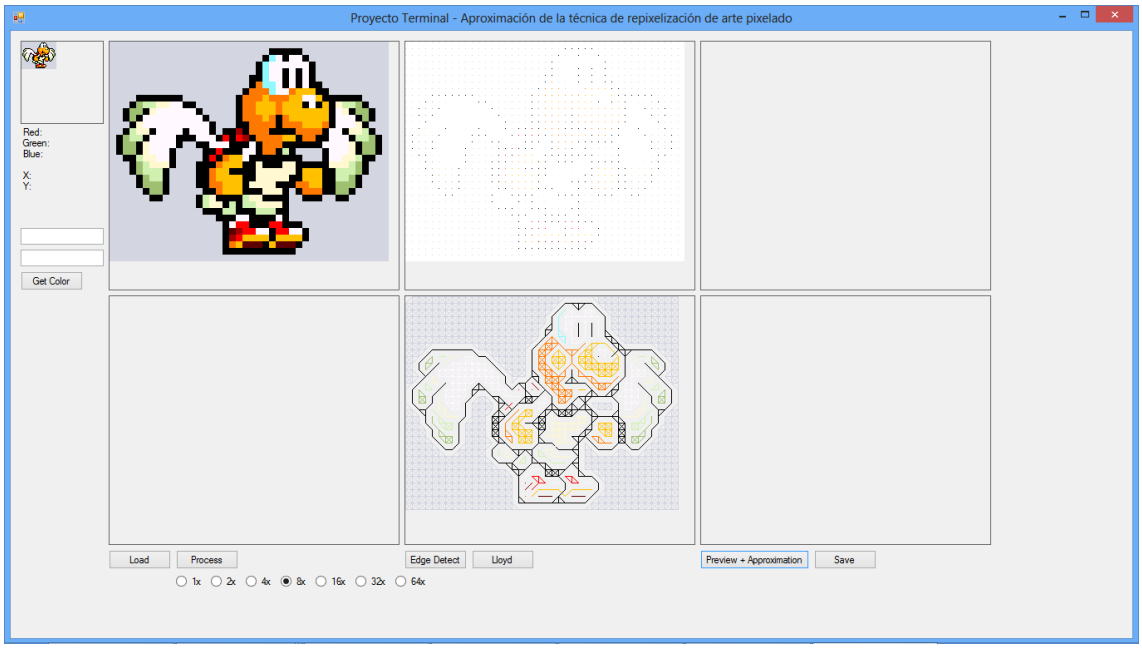


Imagen 3. Interfaz de la aplicación después de mostrar el cálculo del diagrama.

Por último, la aplicación mostrará en el Área 7 la imagen final repixelada.

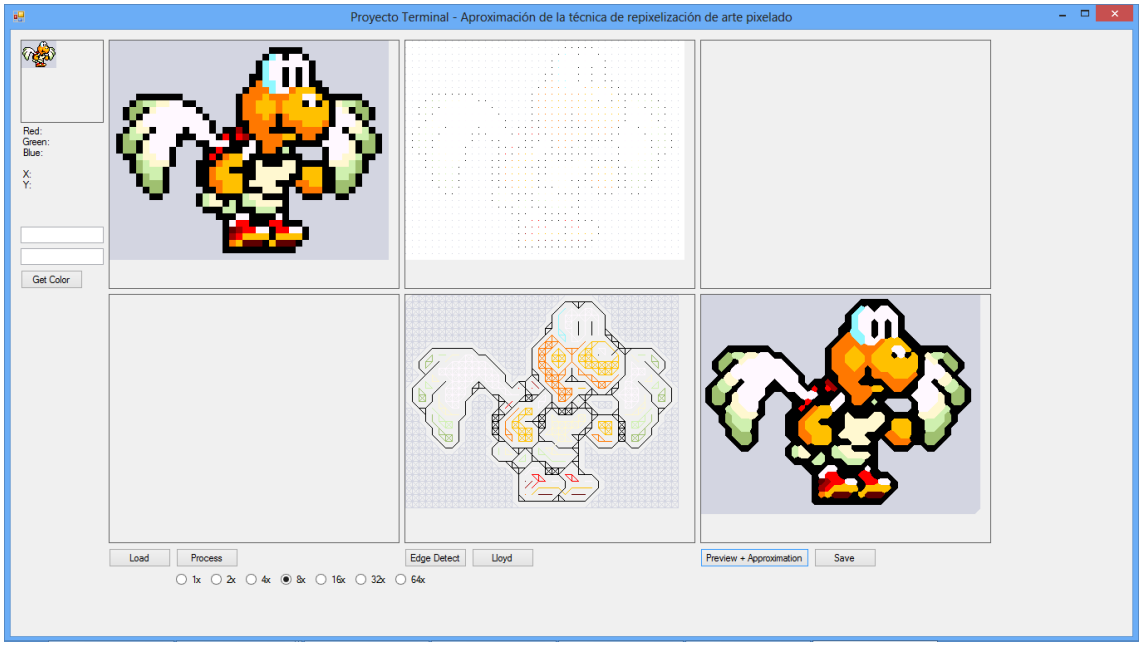


Imagen 4. Interfaz de la aplicación después de mostrar el cálculo de la imagen final.

Finalmente el usuario, si así lo desea, puede guardar una copia de la imagen original y una copia de la imagen final repixelada para poder visualizarlas mejor.

Primero se le preguntará donde desea guardar la imagen original.

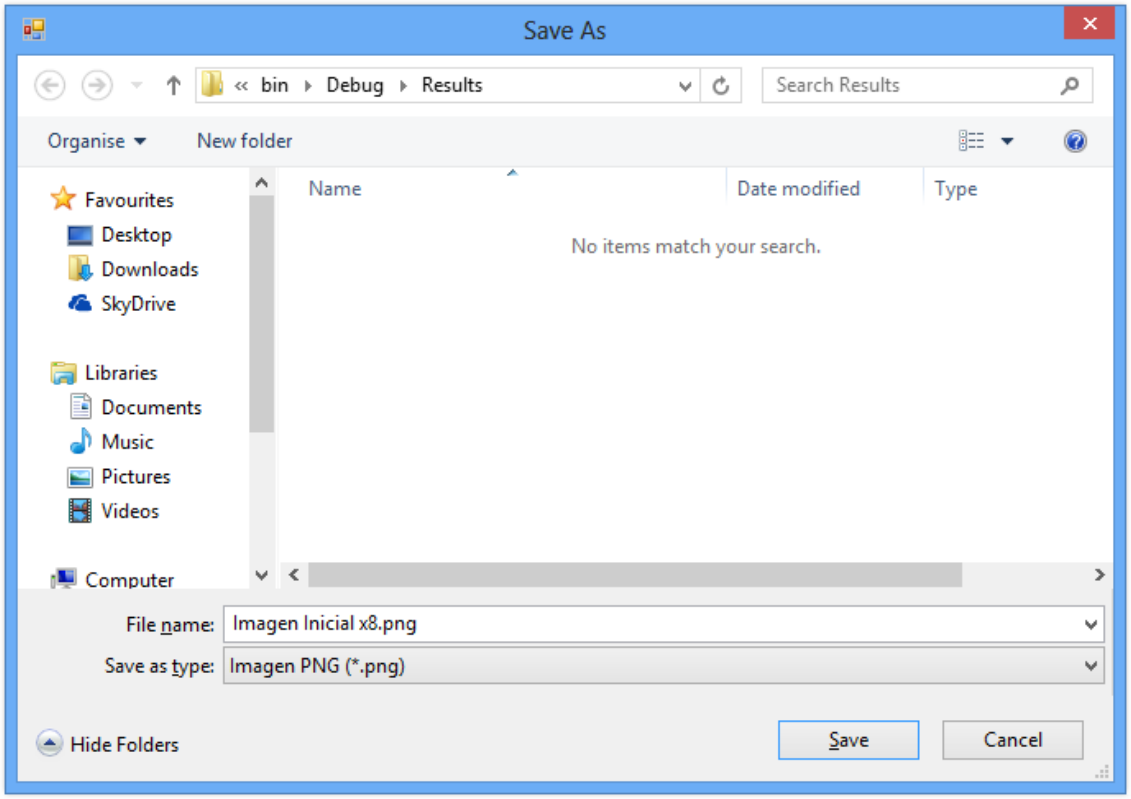


Imagen 5. Interfaz de la aplicación; ventana para elegir donde se guardará la imagen original.

Y después se le preguntará donde desea guardar la imagen final repixelada.

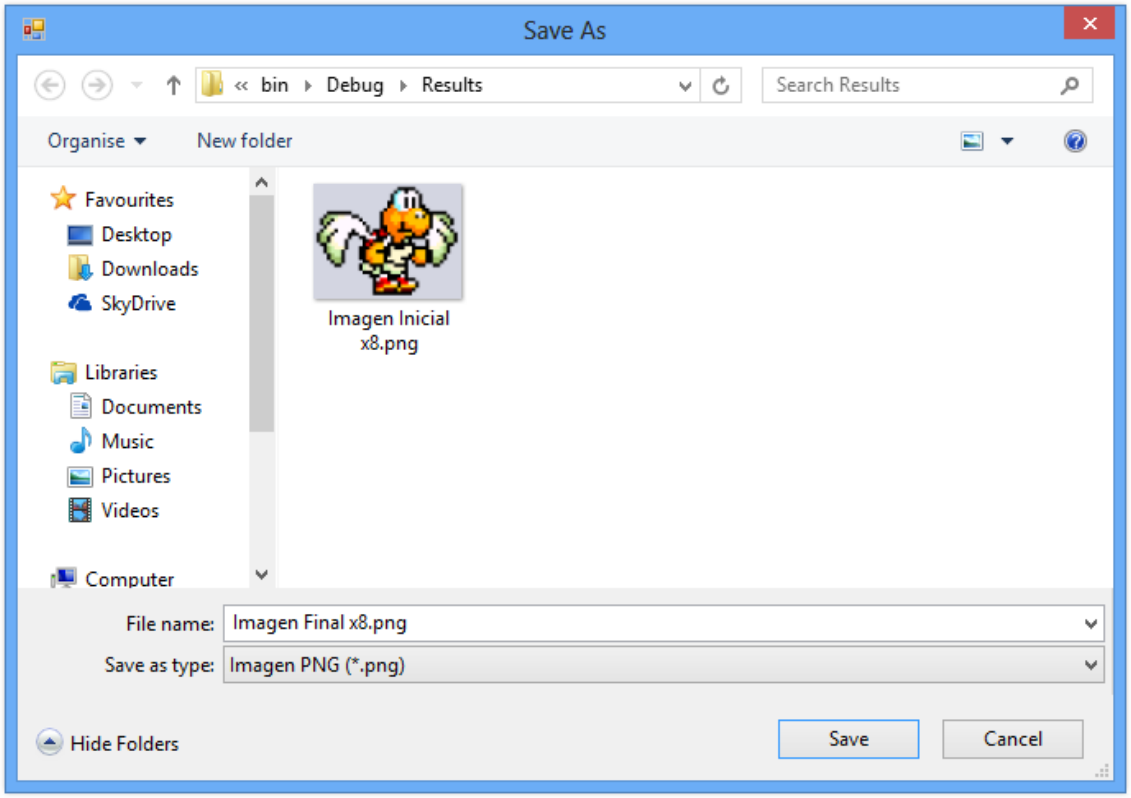


Imagen 6. Interfaz de la aplicación; ventana para elegir donde se guardará la imagen final.