



UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto de Fin de Grado en Ingeniería Informática

Diseño y desarrollo del editor de imágenes UNED Paint

Estudiante: MIHAIL GEORGESCU
Dirigido por: JOSÉ MANUEL DÍAZ MARTÍNEZ
Curso: 2020 - 2021



Diseño y desarrollo del editor de imágenes UNED Paint

Proyecto de Fin de Grado en Ingeniería Informática de modalidad específica

Realizado por: MIHAIL GEORGESCU
Dirigido por: JOSÉ MANUEL DÍAZ MARTÍNEZ

Fecha de lectura y defensa:

Contenido

Resumen	1
Traducción al inglés del título, resumen y palabras clave	3
Listado de siglas, abreviaturas, acrónimos con sus respectivos significados	5
1 Introducción	7
1.1 Motivación del proyecto	7
1.2 Objetivos del proyecto	7
1.3 Base teórica	7
1.4 Estado del arte	9
1.5 Estructura de la memoria	13
2 Análisis	15
2.1 Requisitos funcionales	15
2.2 Requisitos no funcionales	17
2.3 Diagrama de casos de uso	18
2.4 Diagramas de estado	23
3 Diseño	27
3.1 Herramientas empleadas	27
3.2 Godot	28
3.3 Lenguajes de programación utilizados	29
3.4 Arquitectura	30
3.5 Metodología de desarrollo	32
3.6 Organización de los ficheros	36
3.7 Detalles sobre la estructura de las carpetas del proyecto	37
3.8 Espacios de nombre	39

3.9	Mockup interfaz de usuario	40
4	Implementación	43
4.1	La herramienta pincel	43
4.2	Persistencia de los datos	47
4.3	Versionado	47
4.4	Internacionalización	49
4.5	Requisitos hardware y software para la ejecución	50
4.6	Configuración del entorno de desarrollo	51
5	Pruebas	53
5.1	Pruebas automatizadas	53
5.2	Pruebas manuales	56
6	Planificación y presupuesto	59
6.1	Presupuesto	61
6.2	Esfuerzo	61
7	Conclusiones y trabajos futuros	63
7.1	Posibles cambios UNED Paint	63
7.2	Problemas conocidos	64
	Referencias y bibliografía	65
	Apéndice A: Manual de usuario	69

Índice de figuras

1	Ejemplo de imagen r醟ster	8
2	Ejemplo de imagen vectorial en la que se muestran los puntos de control B閦ier	8
3	Krita, pantalla principal	9
4	Photoshop, pantalla principal	10
5	Sketchbook, pantalla principal	11
6	Blender 2.9 en modo pintura de texturas	12
7	Diagrama de casos de uso UNED Paint	18
8	Diagrama de estados del generador de trazos	23
9	Diagrama de estados del pintor	23
10	Diagrama de estados del proceso de importaci髇 de im醕enes	24
11	Diagrama de estados del proceso de exportaci髇 de im醕enes	24
12	Diagrama de estados del proceso de carga de proyectos	25
13	Diagrama de estados del proceso de cambio de la resoluci髇	25
14	Diagrama de estados del proceso de recorte de la imagen	25
15	Diagrama de estados del proceso de desplazamiento de pixeles	26
16	Ejemplo de un 阿rbol de escena en Godot	28
17	La escena Main en el editor Godot	30
18	Una muestra del contenido de la escena ColorPalette.tscn	31
19	UNED Paint en las primeras iteraciones, tratando de hallar como modificar pixeles	32
20	Diagrama de clases, las relaciones del PainterController en Microsoft OneNote	33

21	Diagrama de clases generada automáticamente en Visual Studio, las relaciones del PainterController	34
22	Diagrama de clases del generador de puntos, las relaciones del Brush-StrokeGenerator	35
23	Un ejemplo de organización mixta por relevancia y por tipo	36
24	Las carpetas de UNED Paint	37
25	Organización de los espacios de nombre recomendada	39
26	Organización de los espacios de nombre adoptada en UNED Paint	39
27	Mockup de la interfaz de usuario en Godot Editor	40
28	Sketchbook, pantalla principal	41
29	Pantalla principal de UNED Paint	41
30	Un ejemplo de trazo	43
31	Ejemplo de estampa	43
32	La CPU procesando los paquetes de información de manera secuencial . .	44
33	La GPU procesando los pixeles en paralelo	44
34	Trazo con interpolación	45
35	Trazo sin interpolación	45
36	Secuencia de operaciones simplificada del pintor	45
37	Muestra del shader que utiliza la herramienta pincel	46
38	El script que genera el fichero de versión en Visual Studio 2019	47
39	Git hook que genera el fichero de versión	48
40	Ejemplo de consulta de versión dentro del editor	48
41	Recuperando el string correspondiente a la clave _loadingLabel	49
42	Arquitectura del sistema de traducción	49
43	Muestra del diccionario de traducción	50
44	UNED Paint en el gestor de proyectos del editor Godot	51
45	El proyecto UNED Paint abierto en el editor Godot	52
46	Ejemplo de prueba en la herramienta GDTester	54
47	La carpeta del GDTester	54
48	Las pruebas de UNED Paint en la herramienta experimental GDTester .	55

49	Imagen de prueba creada en UNED Paint	56
50	Imagen de prueba en el editor de imágenes Krita	57
51	Las diferentes regiones de colores y los valores RGBA correspondientes	57
52	Ejemplo de organización de actividades Kanban en Trello	59
53	Una muestra de las "issues" de UNED Paint en GitHub.com	60
54	Diagrama Gantt a grosso modo de las actividades realizadas	60
55	Salario medio mensual para un SOFTWARE DEVELOPER según Salary.com	61
56	Pantalla principal de UNED Paint	69
57	El lienzo	70
58	El panel de información del lienzo	70
59	El selector de herramientas	71
60	El panel de preferencias del pincel	71
61	Ejemplos de trazos con diferentes valores de opacidad y flujo	72
62	Ejemplo de trazo con opacidad 100% y flujo 25%	73
63	Ejemplos de trazos con diferentes valores de suavidad	73
64	Ejemplos de trazos con diferentes valores de espaciado	74
65	El visualizados de trazos	74
66	Resultado de la prueba de velocidad en el panel de notificaciones	74
67	El indicador del pincel sobre diferentes fondos	75
68	Ejemplo de trazado de lineas rectas	76
69	Los botones de activación de la pintura simetrica	77
70	El centro de simetría	77
71	Ejemplo de dibujo simétrico vertical y horizontal	77
72	Imagen antes de utilizar la herramienta de relleno	78
73	Imagen después de utilizar la herramienta de relleno	78
74	La herramienta de muestreo de color	79
75	Dibujando el rectángulo de selección	80
76	Moviendo la selección	80
77	Imagen original antes de recortar	81

78	Dibujando la región de recorte	82
79	La nueva imagen recortada	82
80	El panel de control	83
81	El panel de selección de la resolución	84
82	El panel de control con el indicador de cambios no guardados	84
83	El panel de control sin el indicador de cambios no guardados	85
84	El panel de preferencias	85
85	Los botones de rotación de la imagen	86
86	Imagen antes de realizar la rotación	86
87	Imagen después de realizar la rotación a la derecha	87
88	La paleta de colores	87
89	El selector de color	88
90	El selector de color, selectando un nuevo color	88
91	El selector de color, selectado un nuevo color	88
92	El selector de color, saturación baja	89
93	La capa con el nombre "Dos", dibujada por encima del resto de las capas	90
94	La capa con el nombre "Uno", dibujada por encima del resto de las capas	90
95	Los botones deshacer/rehacer	90
96	El panel de preferencias	91
97	Trazo con interpolación	92
98	Trazo sin interpolación	92
99	Ventana de confirmación de la operación	93
100	Opciones de importación de una imagen	93
101	Opción Escalar la imagen al tamaño del lienzo	94
102	Opción Escalar el lienzo al tamaño de la imagen	94
103	Opción Sin cambios	94
104	El panel de notificaciones	95

Índice de tablas

1	Comparación Ráster y Vector	8
---	-----------------------------	---

Resumen

El proyecto tiene como objetivo crear un editor de imágenes ráster llamado UNED Paint.

El programa tiene implementadas las funcionalidades básicas que esperarías de otros editores modernos ya existentes en el mercado permitiéndonos crear arte digital con cierta fluidez y precisión o simplemente modificar imágenes mediante diversas herramientas de manipulación de los pixeles. Las funcionalidades mas importantes del editor son la herramienta pincel que nos permite hacer dibujos a mano alzada, pintar en capas, importar y exportar imágenes, guardar y recuperar proyectos.

Siendo un editor de imágenes moderno fue imprescindible diseñar la aplicación de tal manera que soporte pantallas táctiles y lápices digitales. Esto se ha conseguido mediante una interfaz de usuario adecuada simplificando el flujo de trabajo del usuario.

Otro aspecto importante que se ha tenido en cuenta en el desarrollo fue la internacionalización y la posibilidad de actualizar con facilidad a otros idiomas. Actualmente el programa está traducido al Inglés y al Español.

El editor se ha creado con la ayuda de varias tecnologías de fuente abierta, siendo la mas destacada el motor de videojuegos Godot, el corazón de UNED Paint.

Palabras clave

GLSL

Editor de imágenes

Sombreador

GPU

Táctil

Godot

Pintar

Ráster

Dibujo a mano alzada

OpenGL

Traducción al inglés del título, resumen y palabras clave

Title

UNED Paint image editor, Design and Development

Abstract

The objective of this project is to create a raster image editor called UNED Paint.

The program has implemented some of the basic functionality that you would expect from other existing modern editors allowing us to create digital art with certain fluidity and precision or to simply modify images using various pixel manipulation tools. The most important features are the brush tool used in freehand painting, image layering, importing & exporting images, saving & loading projects.

Being a modern image editor it was absolutely necessary to design it in such way that it could support touchscreens and digital pens. This was achieved by creating a suitable user interface in order to simplify the workflow.

One other important aspect that we considered is the internationalization and the possibility of adding new languages with ease. Currently the program is translated to English and Spanish.

The editor was created using various open-source technologies the most important one being Godot, the engine that powers UNED Paint.

Keywords

C#
GLSL
Image Editor
Shader
GPU
Touch
Godot
Painting
Raster
Freehand brush
OpenGL

Listado de siglas, abreviaturas, acrónimos con sus respectivos significados

API	Application Programming Interface
CPU	Central Processing Unit
GPU	Graphics Processing Unit
GLSL	OpenGL Shading Language
UI	User Interface
IDE	Integrated Development Environment

1 Introducción

En este documento voy a presentar los aspectos y las decisiones más importantes del desarrollo de UNED Paint.

1.1 Motivación del proyecto

La infinita curiosidad por la informática gráfica y la previa experiencia como usuario con diferentes editores de imágenes me motivaron a desarrollar la herramienta UNED Paint. En este proyecto se juntaron todas estas experiencias y se creó un producto que no pretende ser innovador o diferente, es un problema que se ha explorado ya seguramente miles de veces, pero si se ha intentado crear algo moderno y utilizable por cualquier usuario. Fue un largo viaje de aprendizaje sobre un problema que al principio parecía trivial. Además fue una buena oportunidad para adquirir más conocimientos sobre el mundo de la computación gráfica y sobre la computación en tiempo real, dos campos muy importantes de la informática. Con este proyecto también se ha explorado la posibilidad de desarrollar un producto fiable con software de fuente abierta creado por la comunidad.

1.2 Objetivos del proyecto

El objetivo del proyecto es crear un editor de *imágenes ráster*, con el nombre **UNED Paint**, capaz de realizar las operaciones más comunes y básicas sobre los píxeles, con una interfaz de usuario amigable y ejecutable en la plataforma Windows.

1.3 Base teórica

¿Qué es un editor de imágenes?

Un editor de imágenes es una herramienta software que puede manipular, alterar y crear imágenes digitales. En el mercado existe una cantidad enorme de este tipo de herramientas, de pago y también gratis. Los editores de imágenes pueden tener una multitud de funcionalidades, como añadir filtros a las imágenes, dibujar, cambiar la resolución, recortar, soporte para capas de imágenes, soporte para animaciones, guardar y recuperar proyectos. Esta lista es una pequeña muestra de un sin fin de posibles características.

Otro aspecto importante de los editores es el tipo de imagen que soporta. Las imágenes pueden ser ráster o vectoriales.

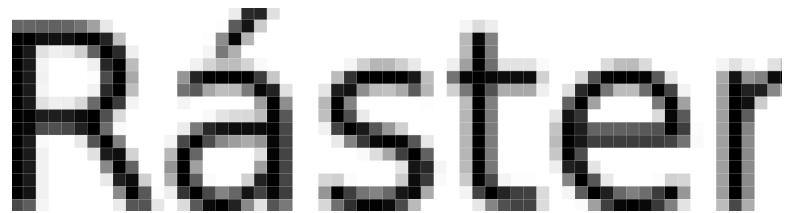


Figura 1: Ejemplo de imagen ráster

Una imagen **ráster**, ver la Figura 1, también conocida como *bitmap image*, es en esencia una matriz de pixeles, en cada celda almacenando información sobre los pixeles correspondientes como el color o el valor de transparencia.

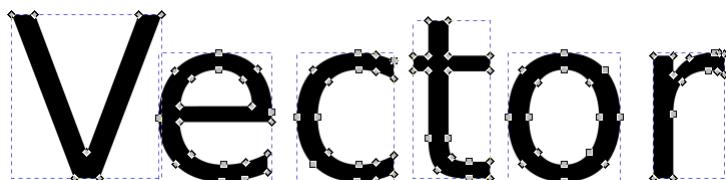


Figura 2: Ejemplo de imagen vectorial en la que se muestran los puntos de control Bézier

En contraste, una imagen **vectorial** ver la Figura 2, se basa en funciones matemáticas para dibujar los gráficos por ejemplo utilizando curvas de Bézier.

Cada uno de estos dos formatos de imagen tiene sus ventajas, desventajas y uso, ver la Tabla 1.

	Ráster	Vector
<i>Uso</i>	Fotografía, Imágenes naturales , etc.	Logos , Icons, etc.
<i>Tamaño archivo</i>	Mayor	Menor
<i>Escalar sin perdida de calidad</i>	No	Si
<i>Facilidad de edición</i>	Si	No

Tabla 1: Comparación Ráster y Vector

Como ya se ha mencionado anteriormente, nuestro objetivo es crear un editor de imágenes **ráster**.

1.4 Estado del arte

En esta sección se van a presentar algunas de las herramientas más populares dedicadas a la edición de imágenes.

Krita

Krita, ver la Figura 3, es un conocido editor de imágenes raster y vectorial de fuente abierta, posiblemente en la misma liga que Photoshop. Es el editor de imágenes en el que más experiencia tengo y fue mi referencia para la funcionalidad implementada en UNED Paint. La herramienta se puede descargar de forma gratuita en la página oficial¹ de ésta.

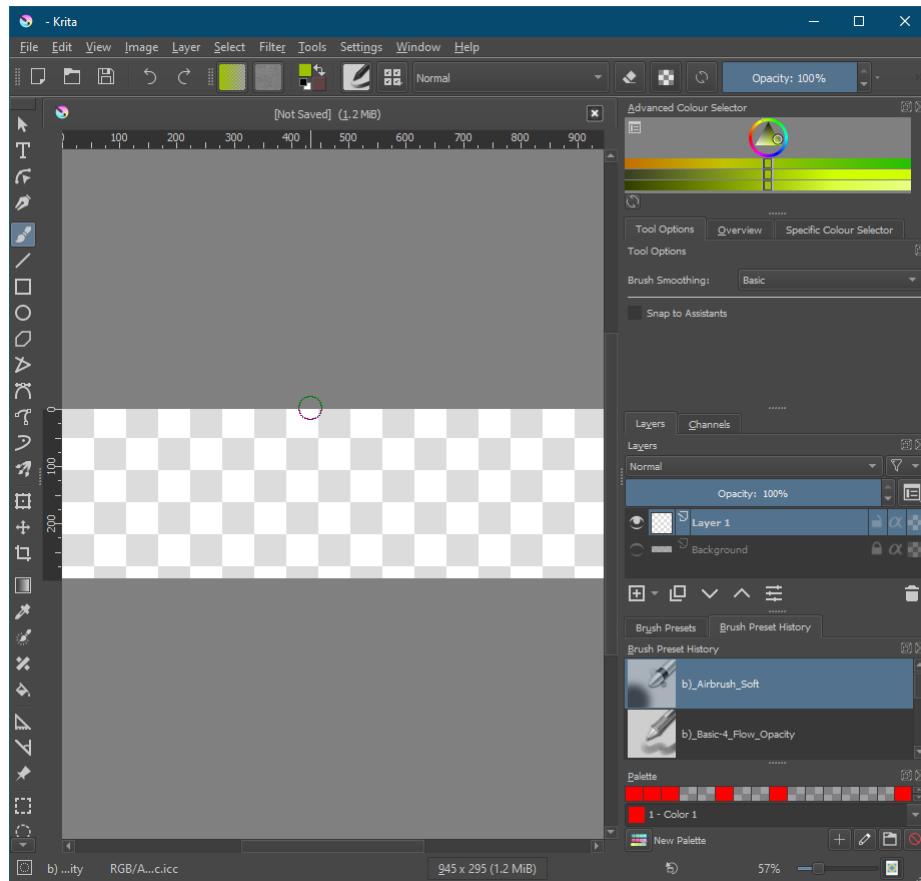


Figura 3: Krita, pantalla principal

¹Página web oficial: www.krita.org

Photoshop

Si hablamos de edición de imágenes, es imposible no mencionar Photoshop.

Photoshop, ver la Figura 4, es uno de los editores de imágenes más populares, creado por Adobe, activamente actualizado con funcionalidad cada vez más compleja, incluso con soporte para modelos 3D. Es una herramienta de pago que se puede comprar en la página oficial² de ésta.

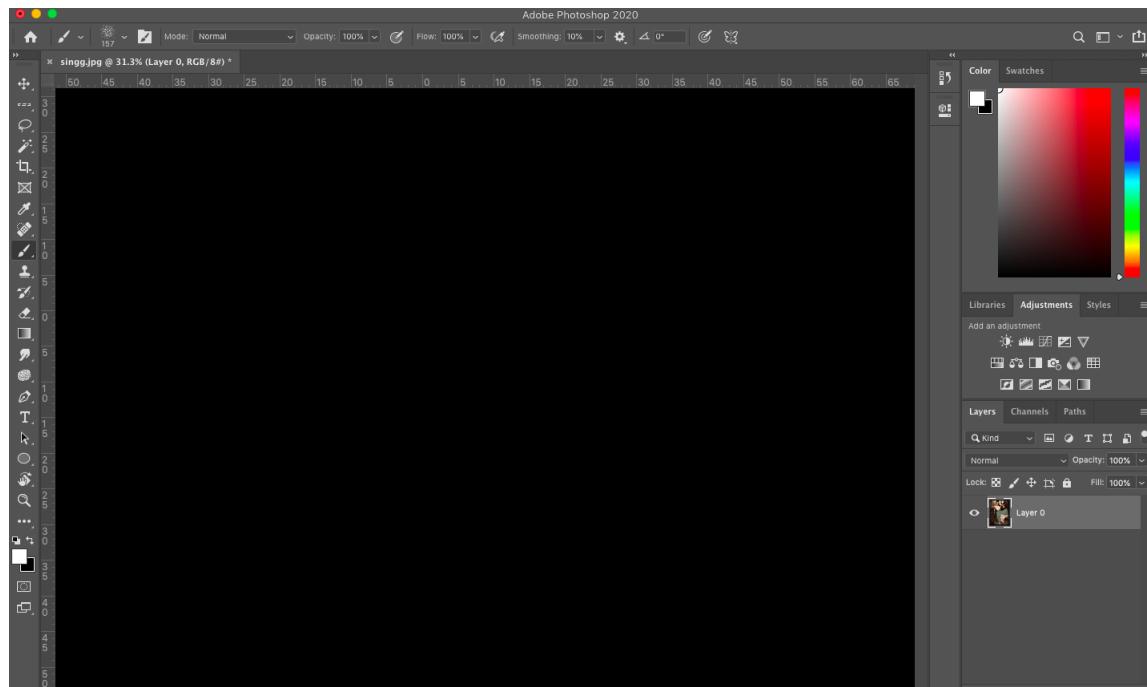


Figura 4: Photoshop, pantalla principal

²Página web oficial: <https://www.adobe.com/>

Sketchbook

Sketchbook³, ver la Figura 5, es un editor de imágenes ráster gratis, creado por Autodesk. Es una herramienta muy ligera, con menos funcionalidad en comparación con las dos mencionadas anteriormente, en principio enfocada en hacer dibujos a mano alzada. Esta aplicación inspiró la interfaz de usuario de UNED Paint.

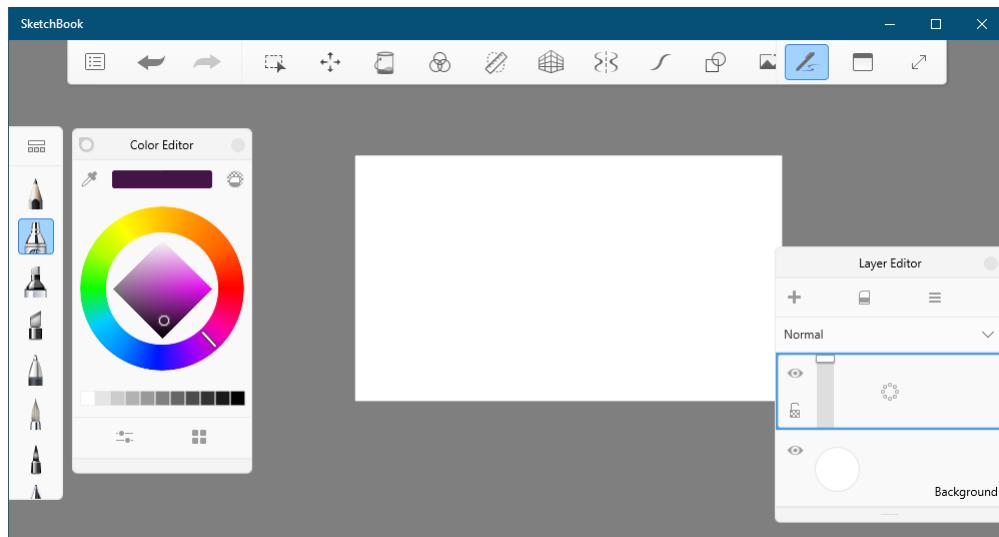


Figura 5: Sketchbook, pantalla principal

³Página web oficial : www.sketchbook.com

Blender

Por último, quiero mencionar Blender, la herramienta de creación de modelos 3D open-source. No es una herramienta dedicada exclusivamente a la edición de imágenes pero si tiene esta funcionalidad, de una forma bastante limitada pero muy interesante. Podemos pintar en 2D y mapear las coordenadas a un modelo 3D, ver la Figura 6. La herramienta se puede descargar de forma gratuita en la página oficial⁴.

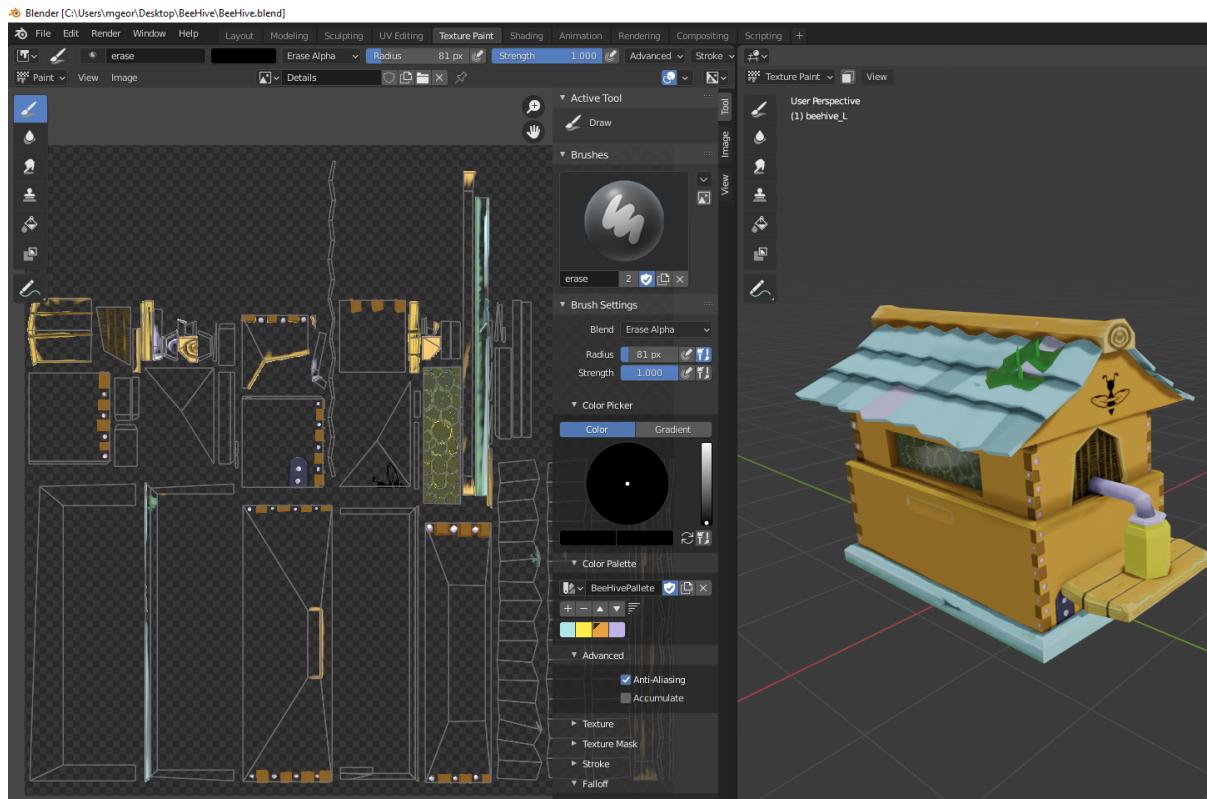


Figura 6: Blender 2.9 en modo pintura de texturas

⁴<https://www.blender.org/>

1.5 Estructura de la memoria

Esta memoria está organizada en los siguientes capítulos:

- 1. Introducción: Pequeña introducción al proyecto como los objetivos y la motivación de este.
- 2. Análisis: En este capítulo se especifican los requisitos del sistema y se presentan los casos de uso.
- 3. Diseño: En este capítulo presentamos las herramientas utilizadas, la metodología seguida a lo largo del proyecto y otras decisiones de diseño relevantes.
- 4. Implementación: Se presentan los aspectos más importantes de la implementación de UNED Paint como problemas y soluciones halladas.
- 5. Pruebas: En este capítulo presentamos las herramientas utilizadas para las pruebas en nuestro proyecto.
- 6. Planificación y presupuesto: En este capítulo presentamos la planificación y calculamos el presupuesto del proyecto UNED Paint.
- 7. Conclusiones: Resumen final con conclusiones y posibles trabajos futuros.

Además se incluye un apéndice con el manual de usuario de la herramienta desarrollada.

2 Análisis

En este capítulo se especificarán los requisitos funcionales, no funcionales y los posibles casos de uso del editor UNED Paint. Al final del capítulo mostramos los diagramas de estado de las principales funcionalidades del programa.

2.1 Requisitos funcionales

RF1. Guardar proyectos en formato propio

- Nos permite guardar las capas, la paleta de colores, resolución del proyecto y otra información necesaria para la reconstrucción de los proyectos en un fichero binario con la extensión .upp (UNED Paint Project).

RF2. Recuperar proyectos previamente creados

- Recuperar y reconstruir los proyectos UNED Paint cargando los ficheros .upp .

RF3. Exportar imágenes

- Exportar imágenes en formato .png preservando la transparencia.

RF4. Importar imágenes

- Cargar imágenes en el proyecto actualmente abierto.
- Crear una nueva capa con la nueva imagen importada.
- Soporte para varios formatos de imágenes como .png, .jpg, .bmp.
- La **resolución máxima** soportada es 4096 x 4096 pixeles.

RF5. Redimensionar el lienzo

- Cambiar la resolución del lienzo y de las imágenes importadas en el proyecto.

RF6. Recortar imágenes

- Seleccionar una región y recortar todas las imágenes importadas en el proyecto.
- Cambiar la resolución del proyecto a la dimensión de la región.

RF7. Herramienta pincel con soporte para la presión del pincel

- Permite hacer dibujos a mano alzada presionando sobre el lienzo.
- El comportamiento de esta herramienta se puede modificar de tal manera que nos permita hacer dibujos simétricos, trazar líneas rectas y borrar píxeles.
- Los parámetros de la herramienta pincel incluyen color, opacidad, radio, flujo, suavidad y espaciado.
- Si el dispositivo que utilizamos dispone de un lápiz con sensor de presión, los datos de presión se pueden utilizar para cambiar dinámicamente el radio o la opacidad de las aplicaciones de color.

RF8. Herramienta de relleno

- Cambia el color muestreado en la posición del lápiz y de todos los píxeles adyacentes similares a un nuevo color elegido.

RF9. Herramienta de muestreo de color

- Permite muestrear el color del lienzo en la posición indicada.

RF10. Herramienta de selección

- Te permite seleccionar y desplazar regiones de píxeles a una nueva posición en el lienzo.

RF11. Previsualizador de trazos

- Nos permite visualizar el aspecto de los trazos de la herramienta pincel en tiempo real y se actualiza con los cambios de los parámetros de ésta.

RF12. Paleta de colores

- Permite guardar colores para su posterior uso en una lista de colores.

RF13. Selector de colores

- Nos permite cambiar el color que utilizará la herramienta activa.

RF14. Gestión de capas de imágenes

- Crear capas.
- Cambiar el nivel de las capas.
- Eliminar capas.
- Duplicar capas.
- Fusionar capas.
- Bloquear capas (imposibilita la modificación de la capa).
- Modificar la visibilidad de las capas.
- Preservar la transparencia (impide la modificación los pixeles completamente transparentes).
- Renombrar las capas.
- Deshacer y rehacer modificaciones en las capas.

RF15. Persistencia de la configuración del editor

- Guardar en el disco información necesaria para la configuración del editor y utilizarla al ejecutar el programa.

2.2 Requisitos no funcionales

RNF1. Internacionalización

- Traducido totalmente al español e inglés.

RNF2. Soporte pantalla táctil y gestos para controlar el lienzo y la interfaz de usuario

RNF3. Interfaz de usuario amigable e intuitiva de utilizar

RNF4. Portabilidad del programa a diferentes plataformas

RNF5. Ejecutable en la plataforma Windows

RNF6. Soporte para los atajos de teclado

- Controlar las diversas herramientas del editor mediante los atajos de teclado.

RNF7. Seguridad

- Para una mayor seguridad crear copias de los ficheros .upp antes de sobrescribirlos, con la extensión .b kp.

2.3 Diagrama de casos de uso

En la Figura 7 podemos observar el diagrama de los casos de uso mas importantes del editor UNED Paint.

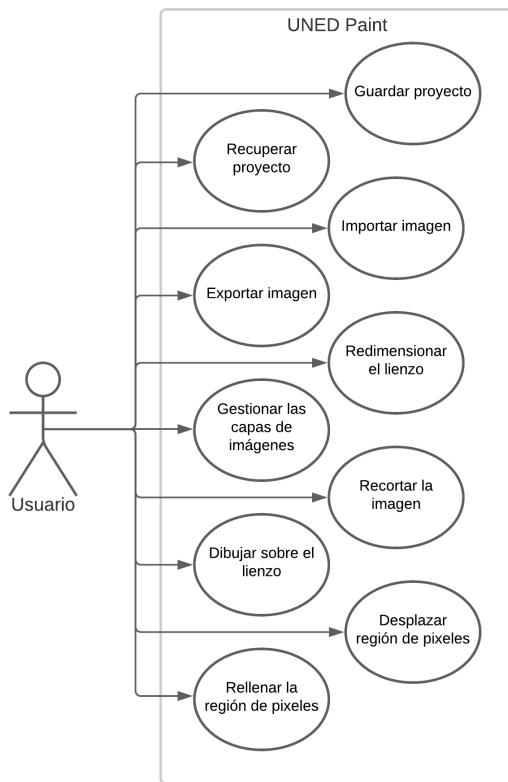


Figura 7: Diagrama de casos de uso UNED Paint

Caso de uso CU1: Guardar proyecto

Actor: Usuario

Flujo normal

1. El Usuario selecciona la operación para guardar el proyecto y facilita al Sistema la dirección.
2. El Sistema comprueba la validez de la dirección.
3. El Sistema recupera la información necesaria, las capas de imágenes y la información de configuración correspondiente a cada capa, la paleta de colores, y guarda toda ésta información en un fichero.

Flujos alternativos

2.1. La dirección introducida no es válida.

Caso de uso CU2: Recuperar proyecto

Actor: Usuario

Flujo normal

1. El Usuario selecciona la operación para recuperar el proyecto y facilita la dirección de éste al Sistema.
2. El Sistema comprueba la validez de la dirección.
3. El Sistema recupera y comprueba la validez del fichero indicado y empieza a configurar el entorno con la información recientemente cargada.

Flujos alternativos

2.1. La dirección introducida no es válida.

2.2. La dirección introducida es válida.

2.2.1. El proyecto actualmente abierto tiene cambios no guardados en el disco, el Sistema pide al usuario confirmar el guardado de éste.

3.1. El Sistema no puede configurar el entorno, los datos cargados no se reconocen.

Caso de uso CU3: Importar imagen

Actor: Usuario

Flujo normal

1. El Usuario selecciona la operación para la importación de una imagen y facilita la dirección de ésta al Sistema.
2. El Sistema comprueba la validez de la dirección indicada.
3. El Sistema pide al Usuario el método de importación.
4. El Usuario le facilita al Sistema el método de importación de la imagen.
5. El Sistema crea una nueva capa con la nueva imagen recientemente cargada y configura el lienzo con el método de importación elegido por el Usuario.

Flujos alternativos

2.1. El Sistema anula la importación, en la dirección indicada no ha encontrado una imagen válida.

3.1. El Usuario no selecciona ningún método de importación y anula la operación.

Caso de uso CU4: Exportar imagen

Actor: Usuario

Flujo normal

1. El Usuario selecciona la operación para la exportación de la imagen y facilita al Sistema la dirección de guardado.
2. El Sistema comprueba la validez de la dirección indicada.
3. El Sistema fusiona todas las capas de imágenes visibles en una única imagen.
4. El Sistema guarda la nueva imagen fusionada en la dirección indicada.

Flujos alternativos

- 2.1. La dirección de guardado no es válida.

Caso de uso CU5: Redimensionar el lienzo

Actor: Usuario

Flujo normal

1. El Usuario selecciona la operación para redimensionar el lienzo y facilita al Sistema la nueva resolución.
2. El Sistema comprueba la validez de la nueva resolución.
3. El Sistema pide al Usuario la confirmación de la operación.
4. El Usuario confirma la operación.
5. El Sistema modifica la resolución del lienzo y de todas las capas de imágenes a la nueva resolución.

Flujos alternativos

- 2.1. La nueva resolución indicada está fuera del rango permitido, el Sistema ajusta el valor de ésta al valor mas cercano válido.
- 4.1 El Usuario cancela la operación.

Caso de uso CU6: Recortar la imagen

Actor: Usuario

Flujo normal

1. El Usuario selecciona la operación para recortar la imagen y facilita al Sistema la dimensión y la posición del recorte.
2. El Sistema comprueba la validez de la dimensión y posición.
3. El Sistema pide al Usuario la confirmación de la operación.
4. El Usuario confirma la operación.

5. El Sistema modifica la resolución del lienzo y de todas las capas de imágenes a la dimensión del recorte en la posición indicada.

Flujos alternativos

- 2.1. La dimensión indicada no es válida.
- 2.2. La posición indicada no es válida.
- 4.1. El Usuario cancela la operación.

Caso de uso CU7: Dibujar sobre el lienzo

Actor: Usuario

Flujo normal

1. El Usuario selecciona la operación para dibujar sobre el lienzo y facilita al Sistema las coordenadas del trazo.
2. El Sistema empieza a modificar los pixeles de la capa actualmente seleccionada en las coordenadas indicadas.

Caso de uso CU8: Deslazar región de pixeles

Actor: Usuario

Flujo normal

1. El Usuario selecciona la operación para desplazar la región de pixeles y facilita al Sistema la dimensión y la posición de la región y la nueva posición desplazada de ésta.
2. El Sistema comprueba la validez de los datos aportados por el Usuario.
3. El Sistema pide al Usuario la confirmación de la operación.
4. El Usuario confirma la operación.
5. El Sistema desplaza la región de pixeles de la capa seleccionada en la nueva posición indicada.

Flujos alternativos

- 2.1. La región indicada no es válida.
- 2.2. La nueva posición indicada no es válida.
- 4.1. El Usuario cancela la operación.

Caso de uso CU8: Rellenar la región de píxeles

Actor: Usuario

Flujo normal

1. El Usuario selecciona la operación para llenar la región de píxeles y facilita al Sistema la posición de relleno y el nuevo color de los píxeles.
2. El Sistema comprueba la validez de la posición de relleno.
3. El Sistema modifica en la posición indicada el color de todos los píxeles adyacentes similares al nuevo color indicado.

Flujos alternativos

- 2.1. La posición no es válida.

2.4 Diagramas de estado

En este apartado podemos observar los diagramas de estado de las funcionalidades principales de UNED Paint.

Vamos a mostrar los estados y las actividades de la herramienta pincel en dos diagramas diferentes. En el diagrama de estados de la Figura 8 podemos observar el proceso de generación de las coordenadas utilizadas por el sistema que se encarga de renderizar los trazos, ver la Figura 9.

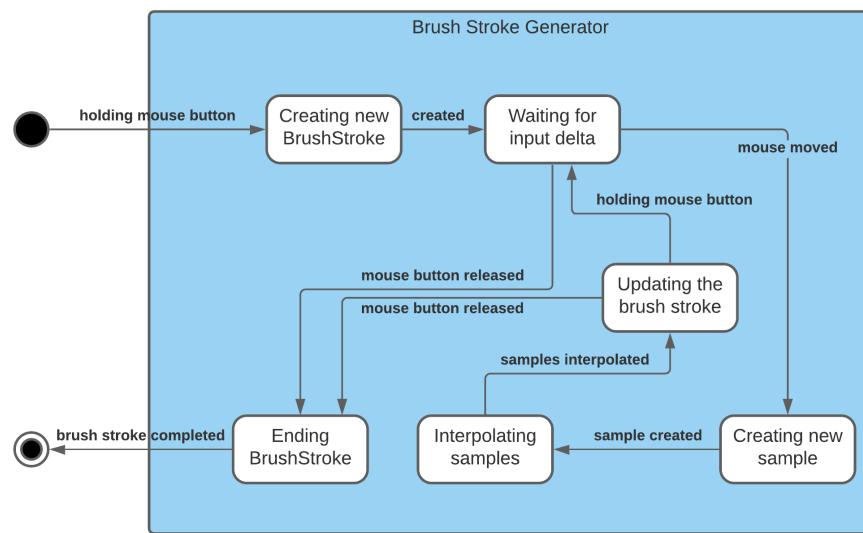


Figura 8: Diagrama de estados del generador de trazos

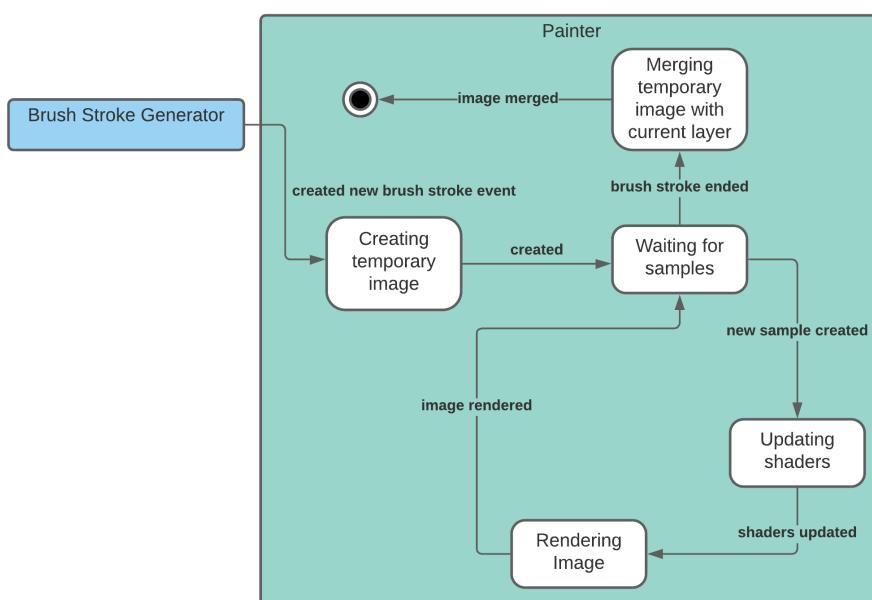


Figura 9: Diagrama de estados del pintor

En la Figura 10 podemos observar los estados del proceso de importación de imágenes. El proceso empieza con la carga de una nueva imagen en el editor y finaliza con la generación de una nueva capa.

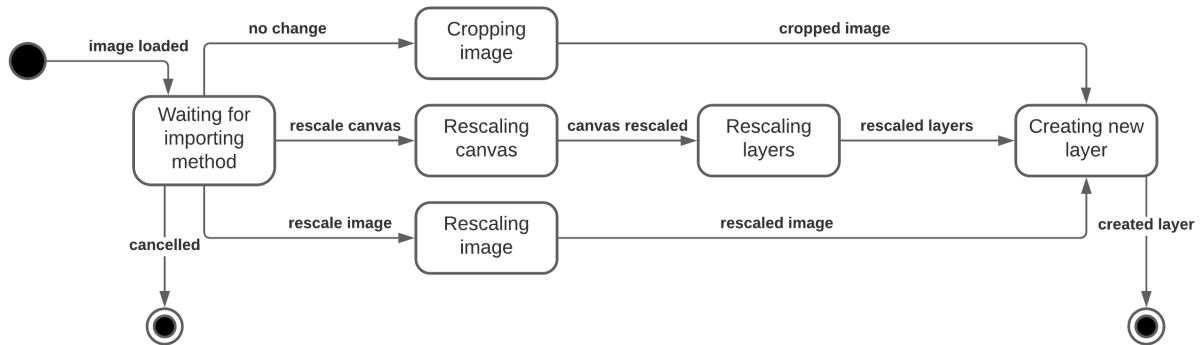


Figura 10: Diagrama de estados del proceso de importación de imágenes

En la Figura 11 mostramos los estados del proceso de exportación de imágenes. En este proceso fusionamos todas las capas de imágenes cargadas en el proyecto y finaliza con el guardado de la nueva imagen generada en el disco.

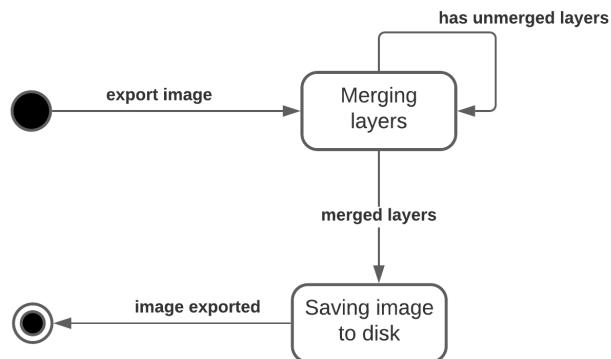


Figura 11: Diagrama de estados del proceso de exportación de imágenes

En el diagrama de la Figura 12 mostramos los estados del proceso de carga de un proyecto.

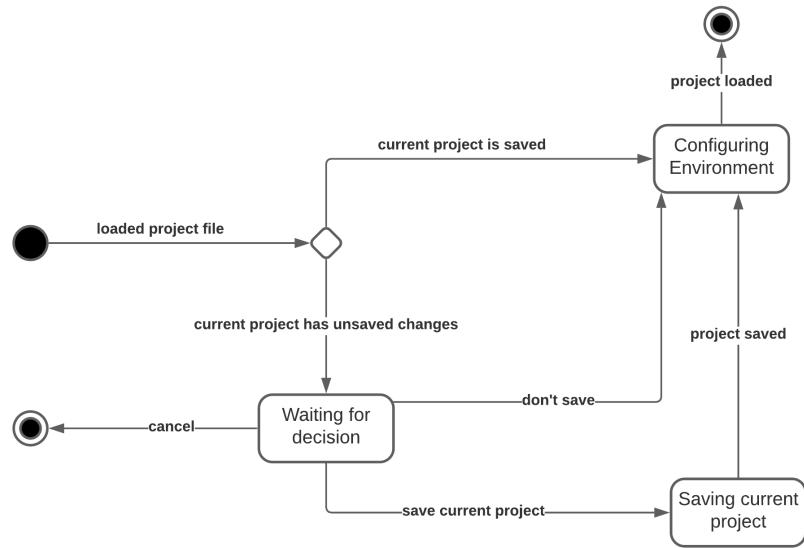


Figura 12: Diagrama de estados del proceso de carga de proyectos

En la Figura 13 mostramos los estados del proceso de cambio de la resolución.

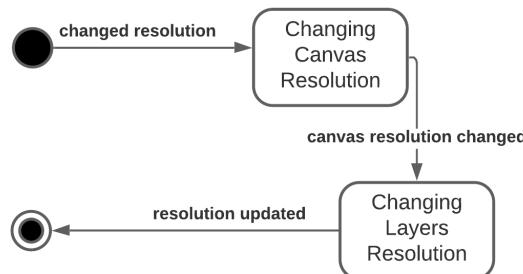


Figura 13: Diagrama de estados del proceso de cambio de la resolución

En la Figura 14 mostramos los estados del proceso de recorte de la imagen.

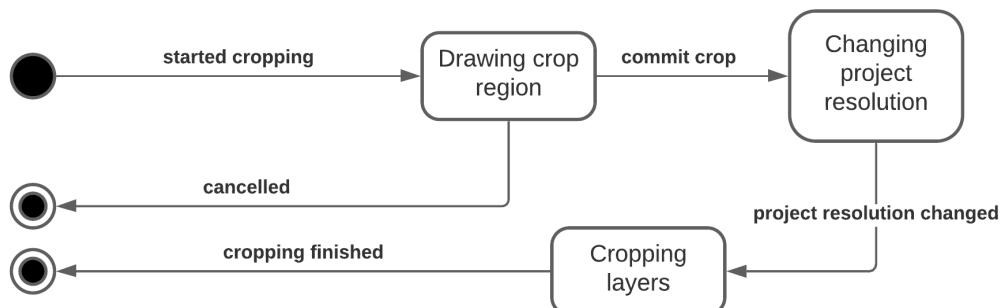


Figura 14: Diagrama de estados del proceso de recorte de la imagen

En la Figura 15 mostramos los estados del proceso de desplazamiento de los píxeles.

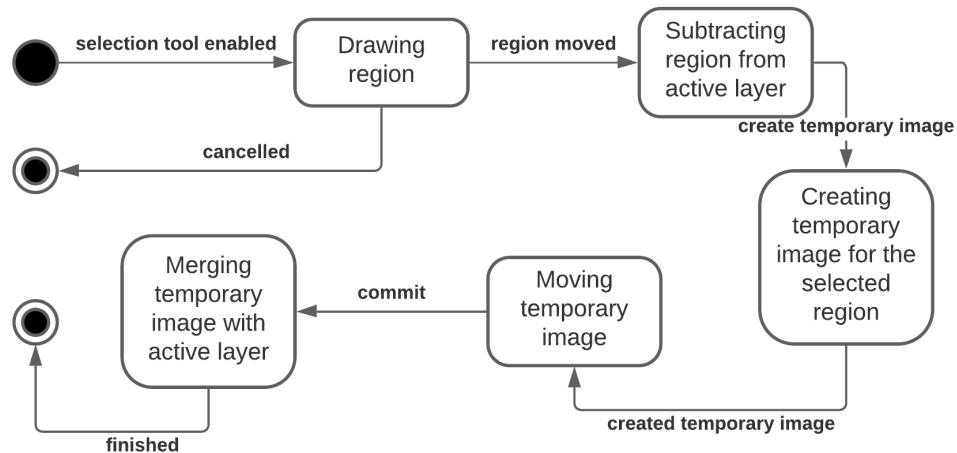


Figura 15: Diagrama de estados del proceso de desplazamiento de píxeles

3 Diseño

3.1 Herramientas empleadas

A continuación voy a enumerar las herramientas más importantes utilizadas en el desarrollo de este proyecto:



Git

Se ha utilizado el sistema de versionado de código fuente Git por ser uno de los más populares y con soporte nativo en Visual Studio 2019 .



GitHub.com

El código fuente se almacenó en un repositorio privado de GitHub.



Godot

Godot, en concreto la *versión 3.2*, es la herramienta principal con la que se desarrolló nuestro editor de imágenes. Más detalles sobre esta decisión en las siguientes secciones.



Inkscape

Es un conocido editor de imágenes vectorial de fuente abierta. Con la ayuda de este se modificaron y crearon los gráficos vectoriales utilizados en UNED Paint.



Krita

Como ya mencionamos en el capítulo 1, Krita fue una de las fuentes de inspiración para la funcionalidad de UNED Paint. Se utilizó para la edición de los gráficos ráster de UNED Paint .



Microsoft Visual Studio 2019

No creo que el IDE Visual Studio necesite una introducción. Es una gran herramienta creada y distribuida por Microsoft sin ningún coste de utilización. Fue una decisión automática puesto que se ha optado programar en C#.



Microsoft OneNote

Es una herramienta creada por Microsoft, útil para tomar notas. Se ha utilizado esta herramienta para crear diagramas rápidamente.

3.2 Godot

Godot⁵ es un motor de videojuegos 2D y 3D de fuente abierta desde el año 2014. Soporta lenguajes de programación como C#, C++, GDScript⁶ y otros. Godot adopta un **diseño orientado a objetos y composición** de escenas.

El editor de Godot es un “videojuego” creado con el motor Godot y facilita la creación de las escenas. Una escena es una composición jerárquica de nodos, formando lo que se conoce como el *árbol de escena*⁷, ver la Figura 16. Se puede extender la funcionalidad básica heredando los nodos.

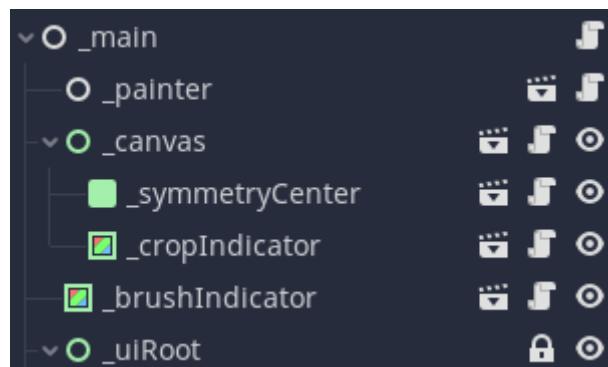


Figura 16: Ejemplo de un árbol de escena en Godot

Elegí Godot por tener un poco de experiencia previa con él y sabía qué era capaz de cumplir los requisitos del editor UNED Paint.

Tenemos a disposición una gran cantidad de librerías y soluciones⁸, posiblemente aún más eficientes y adecuados para facilitar el desarrollo de un editor de imágenes, sin embargo fue una oportunidad perfecta para aprender más sobre lo que ofrece Godot puesto que mi intención es seguir utilizarlo en otros proyectos personales.

⁵www.godotengine.org

⁶lenguaje de programación propio del motor

⁷conocido como SceneTree en Godot

⁸p. ej. Mono Cairo , Microsoft UWP

3.3 Lenguajes de programación utilizados

C#

Es un lenguaje de programación creado por Microsoft con una sintaxis similar a C o C++. UNED Paint es totalmente escrito en este lenguaje.

GLSL

Es un *lenguaje de sombreado* con una sintaxis similar a C creado por OpenGL. Los programas creados en este lenguaje ejecutan en la GPU y tienen un único objetivo, controlar el color de los pixeles de tu pantalla. Godot utiliza un lenguaje de sombreado muy similar a GLSL llamado **Godot Shading Language** con la misma sintaxis y nomenclatura. Se utilizó este lenguaje para la creación de algunos elementos de interfaz de usuario y funcionalidad.

3.4 Arquitectura

Como ya hemos mencionado anteriormente en el capítulo 3.2, Godot fomenta una arquitectura **orientada a objetos** y **composición** de escenas. La funcionalidad y el comportamiento de una escena es determinado por la jerarquía de los nodos hijos. Esto es conocido como el patrón **Composite**⁹.

En nuestro programa la *escena* principal es **Main**, ver Figura 17, que se encuentra en la carpeta **prefabricated**; está compuesta por varias escenas independientes colaborando para cumplir los objetivos de UNED Paint.

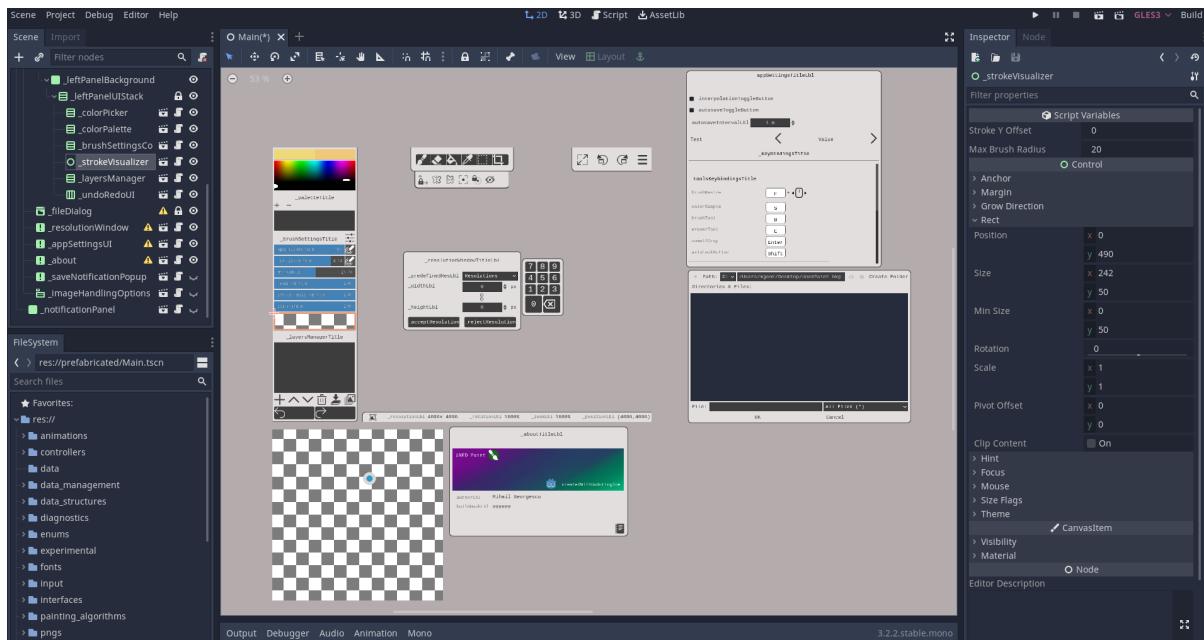


Figura 17: La escena Main en el editor Godot

El **editor Godot** facilita la creación de las **escenas**. Las escenas son ficheros con la extensión **.tscn** y contienen información en formato legible por los humanos como, referencias a recursos, código y valores de parámetros de cada **nodo** de la jerarquía de una escena. Un nodo en si puede ser otra escena. Ver un ejemplo en la Figura 18. Esta información la utiliza el **motor Godot** para instanciar correctamente los objetos al ejecutar.

⁹ https://en.wikipedia.org/wiki/Composite_pattern

```
ColorPalette.tscn [3]
1 [gd_scene load_steps=6 format=2]
2
3 [ext_resource path="res://prefabricated/color_palette/ColorPalette.cs" type="Script" id=1]
4 [ext_resource path="res://themes/White.theme" type="Theme" id=2]
5 [ext_resource path="res://prefabricated/color_palette/plus.svg" type="Texture" id=3]
6 [ext_resource path="res://prefabricated/color_palette/minus.svg" type="Texture" id=4]
7
8
9 [sub_resource type="StyleBoxFlat" id=1]
10 bg_color = Color( 0.243137, 0.239216, 0.239216, 1 )
11
12 [node name="_colorPalette" type="VBoxContainer"]
13 margin_right = 294.0
14 margin_bottom = 116.0
15 script = ExtResource( 1 )
16 __meta__ = {
17     "_edit_use_anchors_": false,
18     "_editor_description_": ""
19 }
20 _elementsPerRow = 7
21 _separation = Vector2( 2, 2 )
22 _highlightColor = Color( 0.811765, 0.207843, 0.207843, 1 )
```

Figura 18: Una muestra del contenido de la escena ColorPalette.tscn

3.5 Metodología de desarrollo

En el anteproyecto se ha planteado un proceso de prototipo desecharable; en realidad se ha empleado un paradigma de **prototipo evolutivo**.

En el **prototipo evolutivo** el equipo de desarrollo implementa rápidamente un prototipo que evoluciona añadiéndole cada vez más funcionalidad y así hasta que se consigue el producto final. Habitualmente el desarrollo empieza por los requisitos menos entendidos. El prototipo evolutivo es similar al desarrollo incremental.

Como no conocíamos muy bien el problema, en las primeras iteraciones del prototipo, ver la Figura 19, el trabajo se ha enfocado en el objetivo principal, conseguir manipular de alguna forma los pixeles de una imagen. En las subsecuentes iteraciones se implementaron y mejoraron progresivamente las diversas funcionalidades del editor basándonos en la experiencia obtenida.

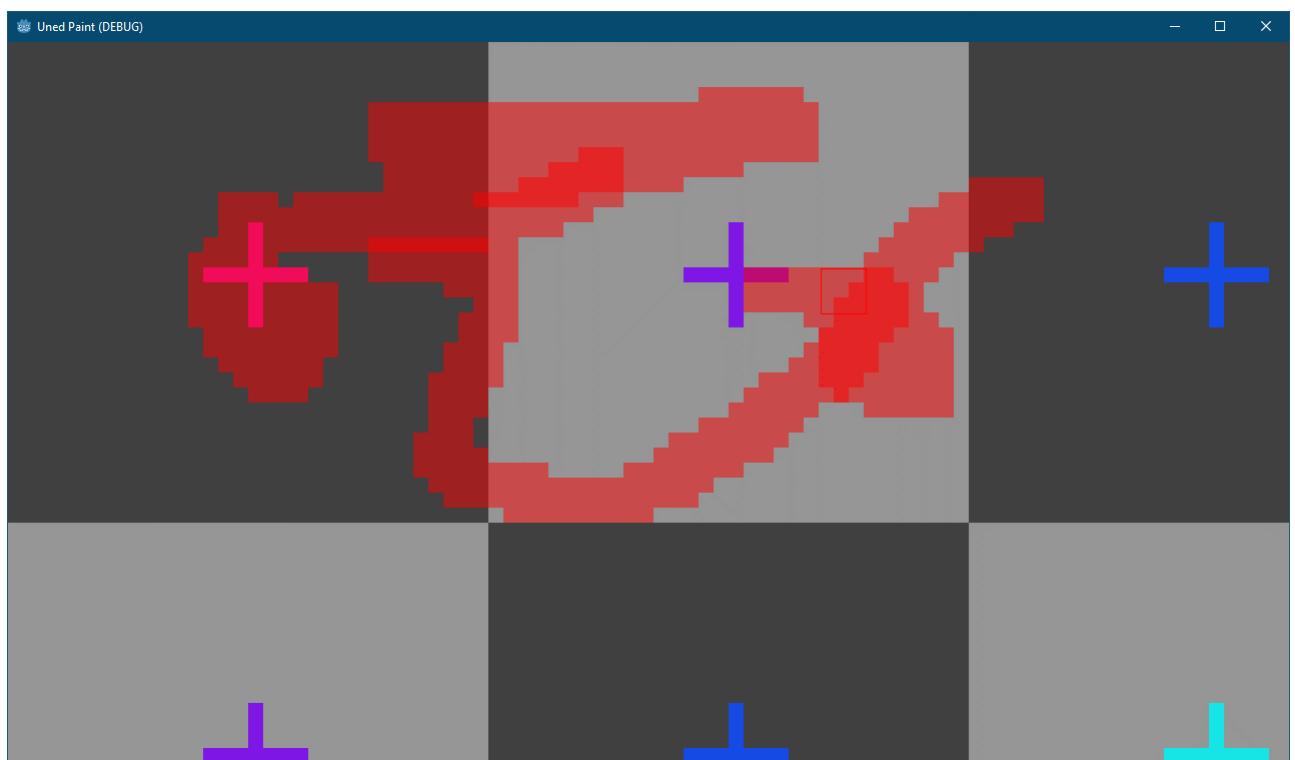


Figura 19: UNED Paint en las primeras iteraciones, tratando de hallar como modificar pixeles

Para agilizar el proceso de desarrollo, el diseño se ha realizado en **Microsoft OneNote**, con esbozos rápidos suficientes para entender el problema, ver un ejemplo en la Figura 20.

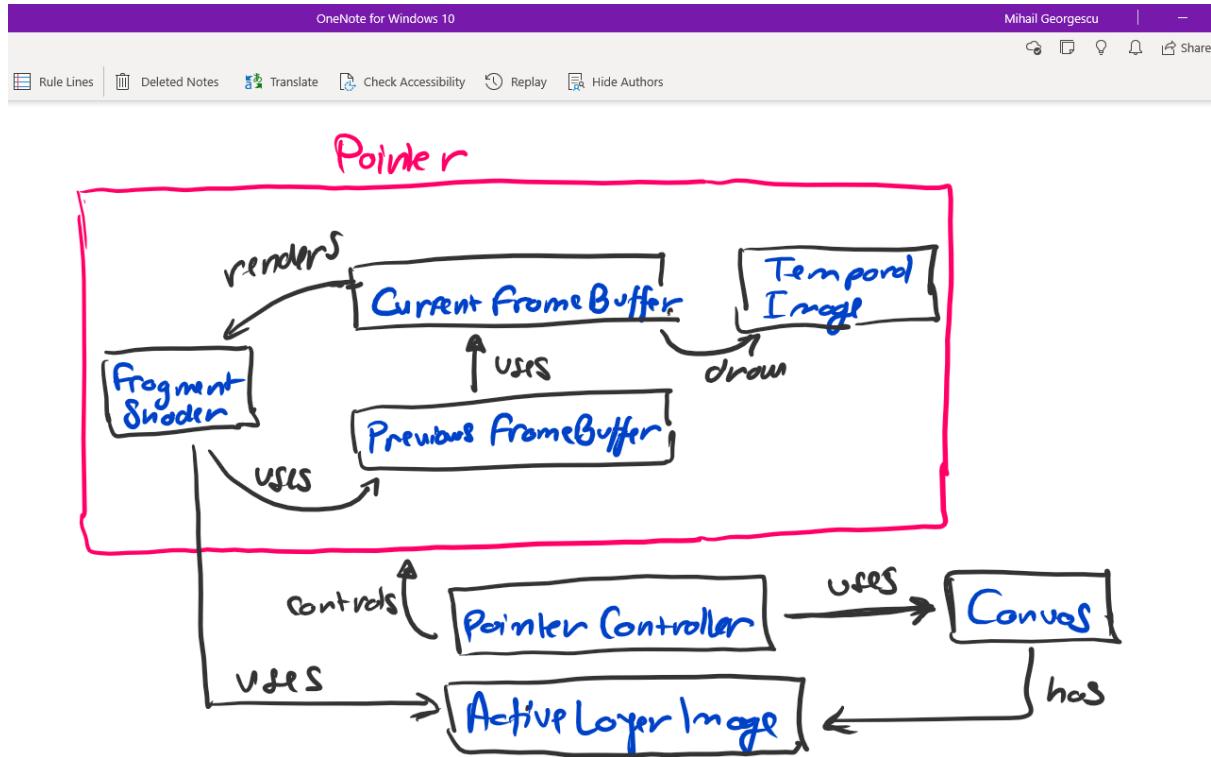


Figura 20: Diagrama de clases, las relaciones del PainterController en Microsoft OneNote

Para generar los diagramas de clases y entender las diferentes relaciones del sistema, podemos utilizar la herramienta de *ingeniería inversa de código hacia diagramas* de Visual Studio, observar los ejemplos en la Figura 21 y 22.

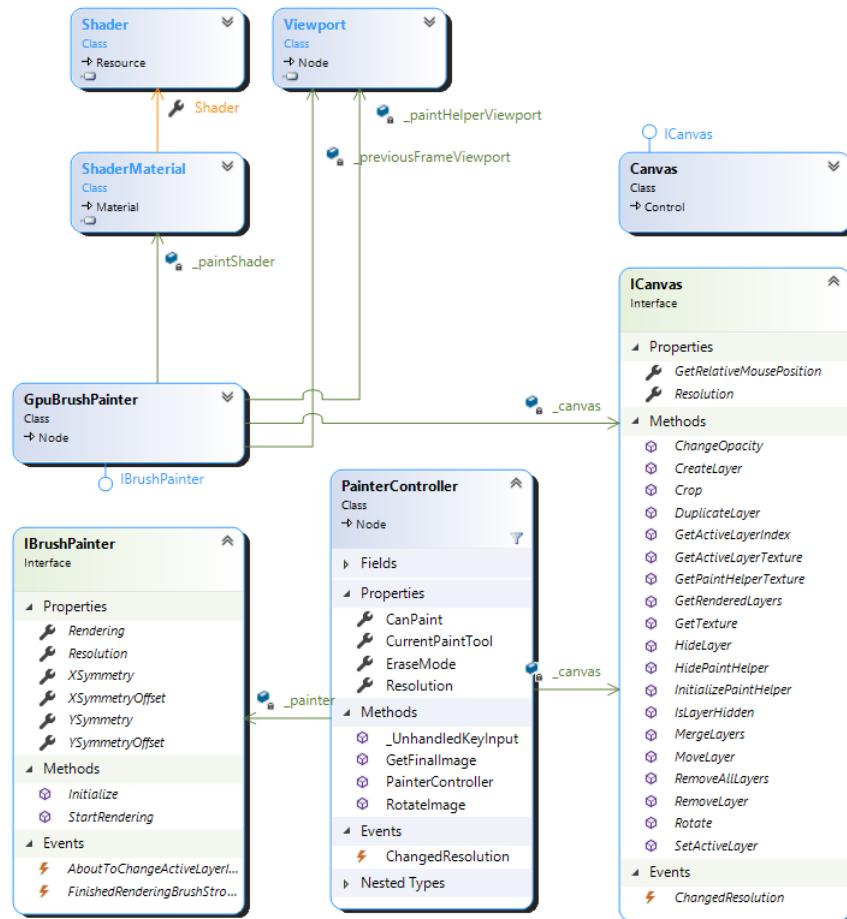


Figura 21: Diagrama de clases generada automáticamente en Visual Studio, las relaciones del PainterController

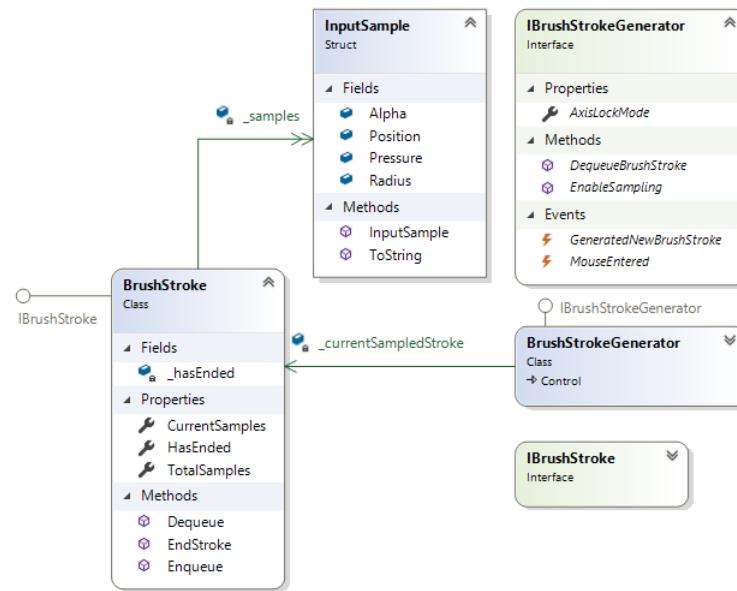


Figura 22: Diagrama de clases del generador de puntos, las relaciones del BrushStrokeGenerator

3.6 Organización de los ficheros

La organización de los ficheros del proyecto es un tema muy importante. Una organización adecuada y consistente simplificará el flujo del trabajo y nos permitirá localizar y acceder con rapidez a cualquier recurso .

Se ha decidido organizar los ficheros por **relevancia** y por **tipo** (categoría). En este proyecto utilizamos una gran variedad de recursos como imágenes (.png, .svg), sombreadores(.shader), escenas(.tscn), código(.cs), fuentes (.ttf) etc. Los recursos estrechamente relacionados son guardados lo más cerca posible uno del otro, esto es lo que llamamos organización por **relevancia**. En la Figura 23 mostramos una posible organización siguiendo esta lógica.

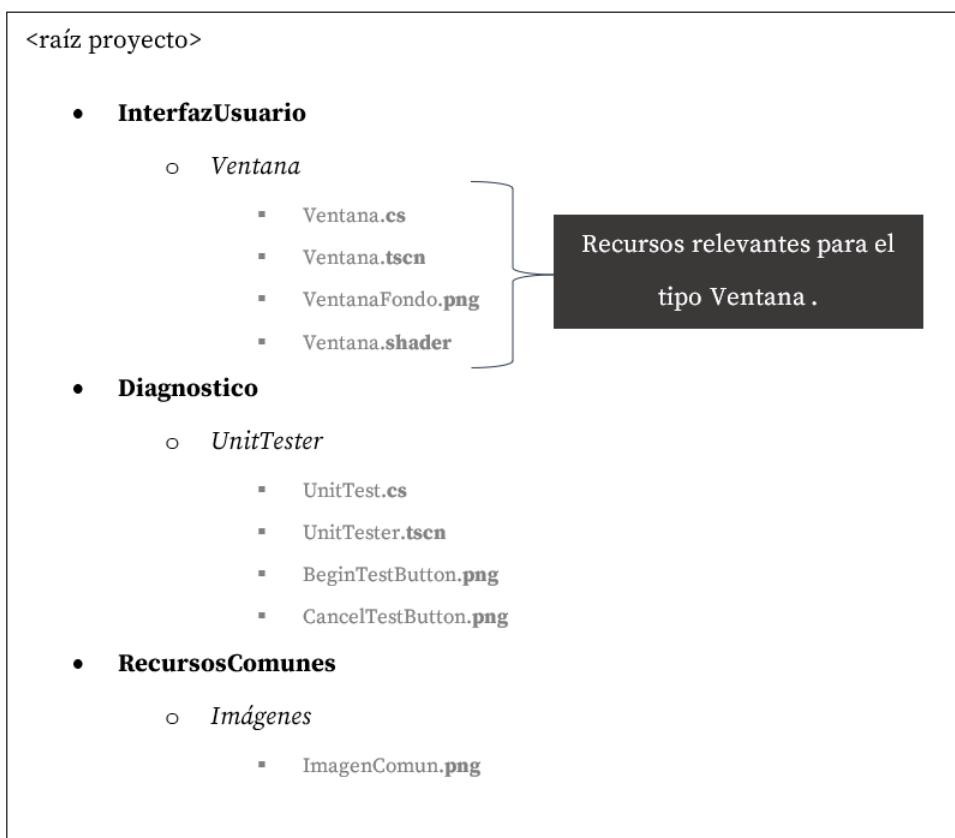


Figura 23: Un ejemplo de organización mixta por relevancia y por tipo

En la Figura 23 tenemos tres grandes categorías **UI**, **Diagnóstico** y **RecursosComunes**. En la carpeta **Ventana** almacenamos todo lo que es relevante para este tipo. Si varios tipos utilizan un recurso común este se colocará en una carpeta común a un nivel superior.

3.7 Detalles sobre la estructura de las carpetas del proyecto

En la Figura 24 puedes observar la estructura de las carpetas del proyecto UNED Paint.

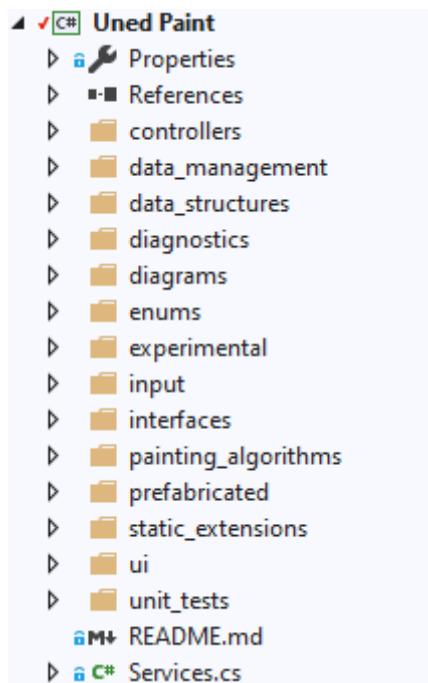


Figura 24: Las carpetas de UNED Paint

A continuación explicamos brevemente el propósito de cada una de las carpetas.

controllers

- Alberga código relacionado con los diferentes controladores.

data_management

- Alberga el código relacionado con los gestores de datos.

data_structures

- Alberga el código de las estructuras de datos utilizadas.

diagnostics

- Alberga el código que se encarga de realizar el diagnóstico de la aplicación como por ejemplo los sistemas que realizan las pruebas unitarias.

enums

- En esta carpeta encuentras la mayoría de los enum utilizados en el proyecto.

experimental

- En esta carpeta encuentras experimentos, sistemas que todavía no se integraron en la aplicación.

input

- En esta carpeta encuentras código que se encarga de capturar la entrada del usuario.

interfaces

- Alberga las diferentes interfaces utilizadas en el proyecto.

painting_algorithms

- En esta carpeta encuentras algoritmos relacionados con la pintura.

prefabricated

- Alberga las escenas Godot y sus correspondientes recursos como código, imágenes etc.

static_extensions

- En esta carpeta encuentras las *extensiones estáticas*¹⁰ de tipos.

ui

- Alberga código relacionado con la interfaz de usuario.

unit_tests

- Alberga las pruebas unitarias.

¹⁰<https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/classes-and-structs/extension-methods>

3.8 Espacios de nombre

En este apartado vamos a explicar las decisiones tomadas sobre los espacios de nombre en nuestro proyecto.

Habitualmente se recomienda que los espacios de nombre correspondan a las carpetas del proyecto, ver el ejemplo de la Figura 25. Considero que es una estructura muy rígida y no nos permite realizar con facilidad modificaciones de las carpetas o reorganizar las clases del proyecto. Cualquier modificación en esta estructura implica un cambio en los espacios de nombre y en las referencias a éstos. Es un trabajo enorme en un entorno que pueda cambiar con frecuencia. Por este motivo se ha decidido no tener ninguna correspondencia entre las carpetas del proyecto y los espacios de nombre. Todas las clases del proyecto están organizadas en un único espacio de nombres llamado UnedPaint, ver la Figura 26.

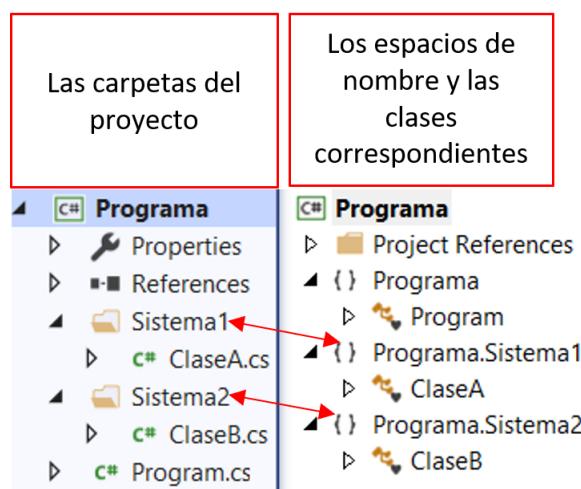


Figura 25: Organización de los espacios de nombre recomendada

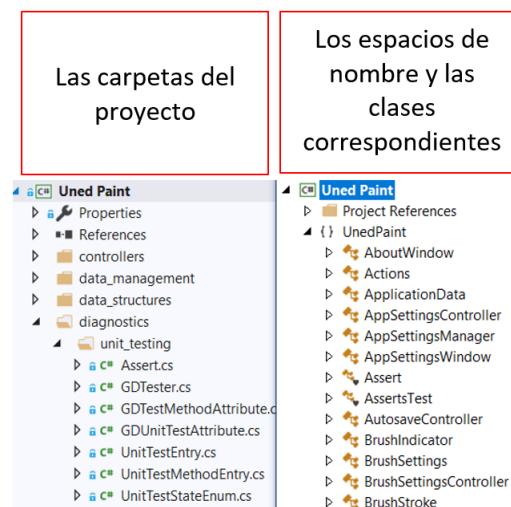


Figura 26: Organización de los espacios de nombre adoptada en UNED Paint

3.9 Mockup interfaz de usuario

En la Figura 27, podemos observar el mockup de la interfaz de usuario de UNED Paint. Se ha concebido una UI con las siguientes características:

- Compatible con una pantalla táctil, *touch friendly*¹¹.
- Sencilla y sin distracciones.
- Intuitiva.

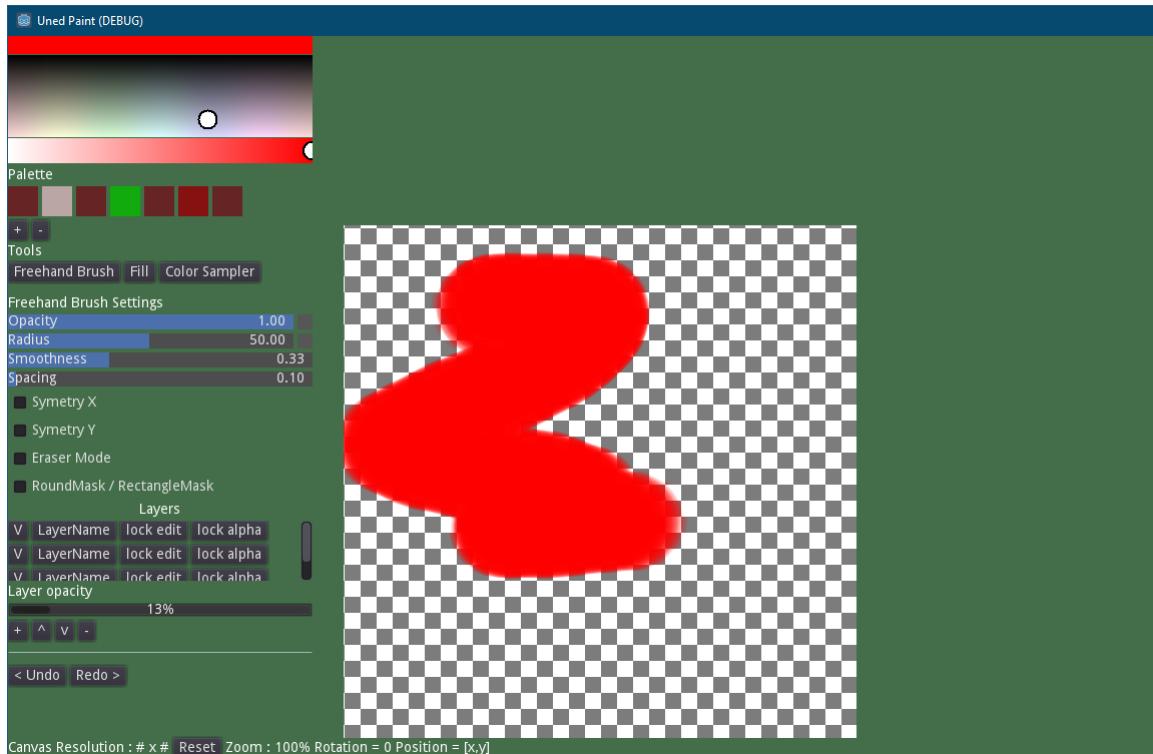


Figura 27: Mockup de la interfaz de usuario en Godot Editor

¹¹ Una posible definición para la interfaz de usuario touch friendly: Elementos de interfaz de usuario grandes, compatibles con una pantalla táctil y con un flujo de trabajo simplificado.

Mi inspiración y referencia principal de la UI fue el editor de imágenes Sketchbook, ver la Figura 28. El resultado final se puede observar en la Figura 29.

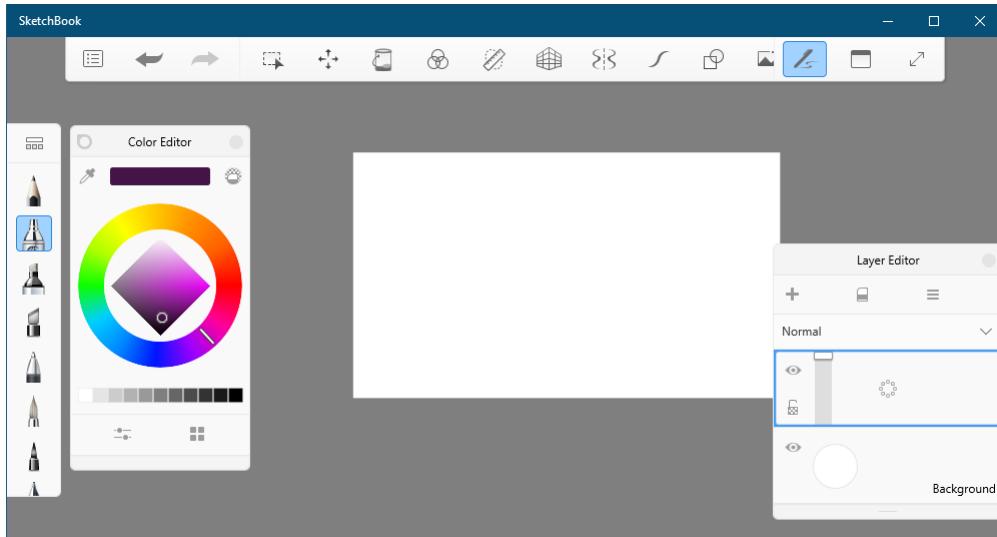


Figura 28: Sketchbook, pantalla principal

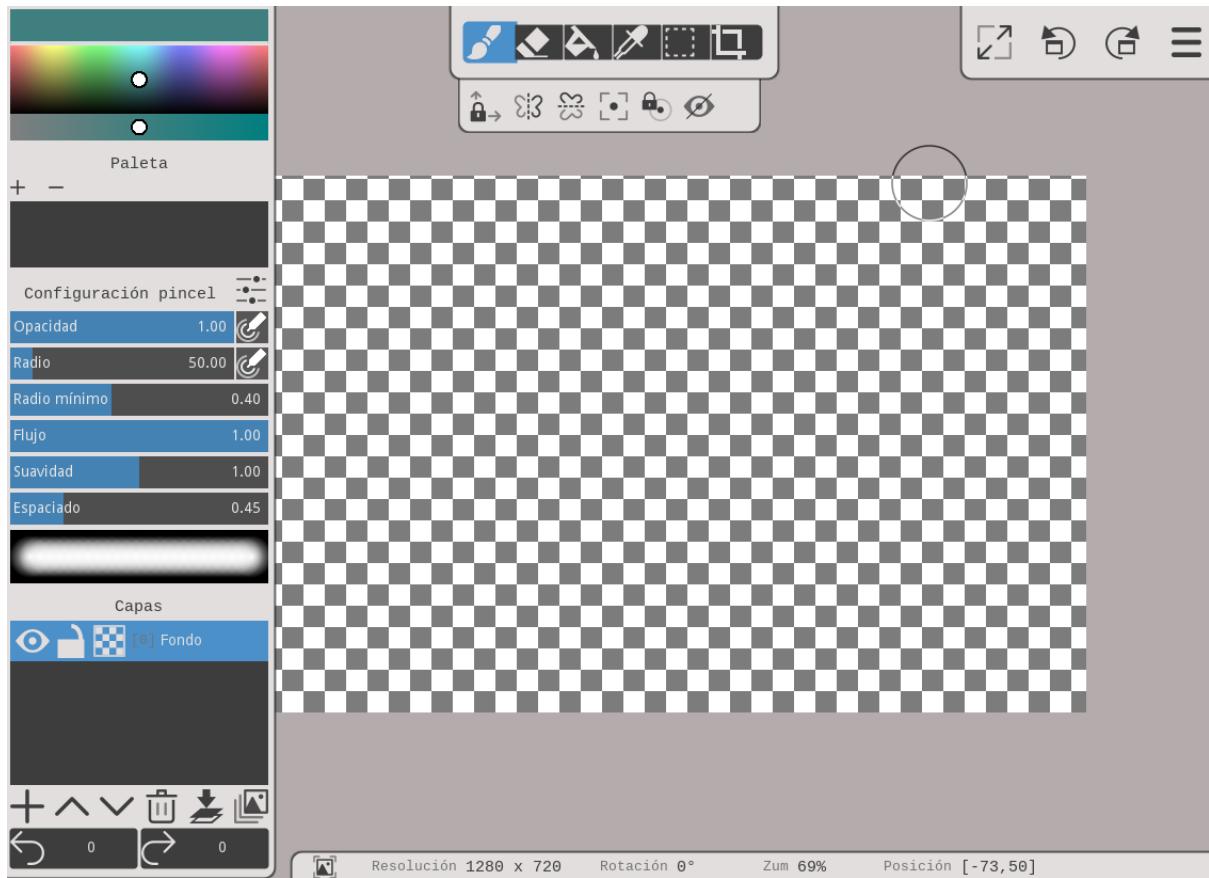


Figura 29: Pantalla principal de UNED Paint

4 Implementación

En este capítulo vamos a presentar las decisiones de implementación más importantes que hemos tomado.

4.1 La herramienta pincel

La herramienta pincel se utiliza para dibujar trazos, ver Figura 30, mediante una técnica de “estampado”, en cada posición del ratón dibujando una estampa, ver la Figura 31. Un trazo está compuesto por varias estampas. En nuestro caso una estampa es un simple círculo relleno aunque podría tener cualquier otra forma o textura.



Figura 30: Un ejemplo de trazo

El problema es el siguiente: ¿cómo rellenamos este círculo lo más rápido posible, cientos de veces en tiempo real?

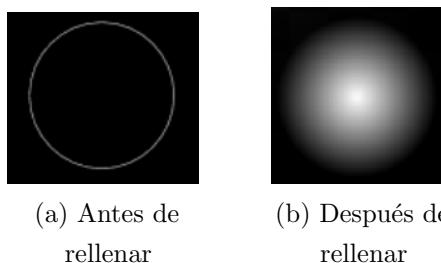


Figura 31: Ejemplo de estampa

Sabía que la utilización de la herramienta pincel tenía que ser en tiempo real, que el trazo tenía que actualizarse milisegundos. En las primeras versiones intenté manipular los pixeles en puro código C#, ejecutado por la CPU; resultó extremadamente lento ya que la CPU procesaba cada píxel de forma secuencial, observar en la Figura 32 una analogía interesante.

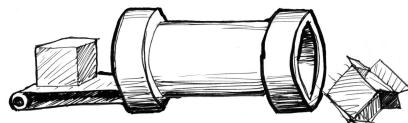


Figura 32: La CPU procesando los paquetes de información de manera secuencial

La mejor opción es manipular los píxeles en la GPU utilizando un sombreador (shader); por este motivo se ha tenido que aprender los básicos del lenguaje **GLSL**. La diferencia de rendimiento entre la versión CPU y la GPU es enorme. Evidentemente este resultado dependerá de muchas variables, versión de Godot, lenguaje de programación, API gráfico, hardware etc; por ejemplo una imagen con resolución 1920 x 1080 (2 073600 píxeles) se puede procesar en la CPU en un tiempo estimado de 100ms, esto es extremadamente lento para una aplicación en la que se pretende dibujar en tiempo real; este resultado a excluido cualquier otro tipo de procesamiento, nuestra estampa será mucho más compleja computacional que llenar un simple rectángulo. Una estampa tendrá varias características como diámetro, suavidad, opacidad, color y otras.

En una GPU moderna tenemos miles de núcleos dedicados exclusivamente al procesamiento de los píxeles ¿porque no explotar esto? Utilizando un sombreador de píxeles podemos dibujar una estampa, dependiendo de la GPU, en menos de un milisegundo con independencia de la resolución de esta, ya que la información de cada píxel se procesa en paralelo, observar en la Figura 33 una analogía interesante.

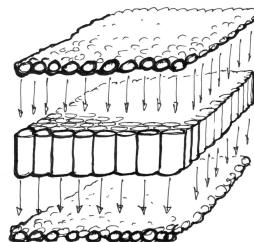


Figura 33: La GPU procesando los píxeles en paralelo

Para realizar pruebas de velocidad de la herramienta pincel se ha implementado una herramienta dedicada que se presenta en el Manual de Usuario A.6 El visualizador de trazos.

Uno de los cuellos de botella más destacados en la implementación actual es que la interpolación del ratón se computa en la CPU (interpolamos la posición y presión aplicada sobre el lienzo), siendo necesario actualizar el sombreador cientos de veces en un segundo. El algoritmo de interpolación es suficientemente rápido, el tiempo que necesita es despreciable, pero actualizar el sombreador es muy lento. Lo ideal sería implementar un

sombreador que realice la interpolación de los puntos directamente en la GPU por ende actualizar el sombreador menos veces. Necesitamos la interpolación porque él procesador no puede actualizar con tanta granularidad las posiciones del ratón. Sin la interpolación nuestro trazo tendría huecos, ver un ejemplo de trazo con interpolación y sin interpolación en la Figura 34 y 35.

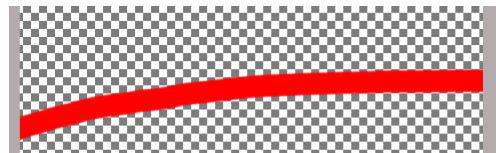


Figura 34: Trazo con interpolación



Figura 35: Trazo sin interpolación

En la Figura 36 mostramos una visión general de las operaciones que realiza la herramienta pincel.

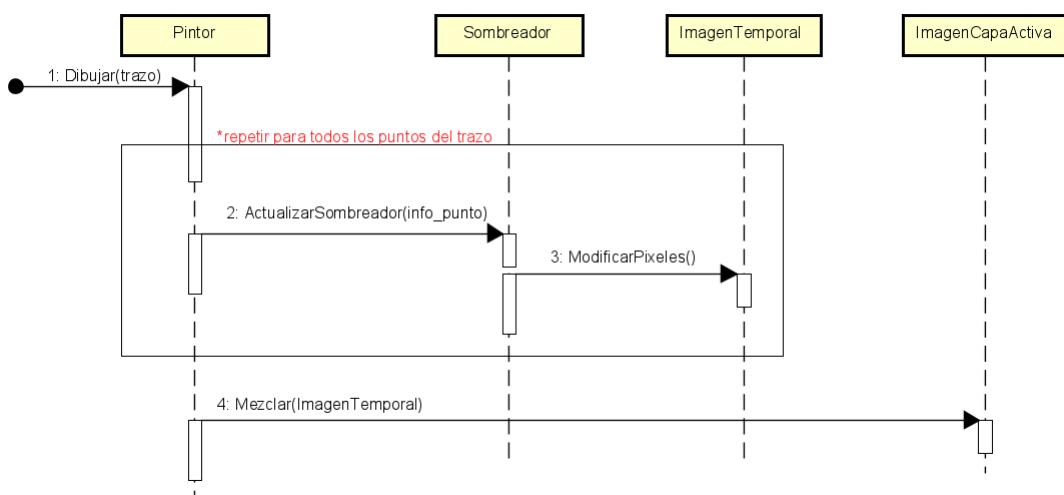


Figura 36: Secuencia de operaciones simplificada del pintor

Una muestra del código del sombreador de la herramienta pincel se puede observar en la Figura 37. La sintaxis es similar a la del lenguaje de programación C. Lo que hace más difícil programar shaders es que tenemos que pensar de una manera muy diferente a lo que estamos acostumbrados. Cada pixel (en realidad **fragmento**¹²) ejecuta en paralelo

¹²https://www.khronos.org/opengl/wiki/Fragment_Shader

el código del shader con los mismos datos. El resultado de la ejecución no es nada mas que el color final del pixel correspondiente. Para dificultar aún más la programación de los shaders, estos no pueden acceder a la información de la ejecución previa, al menos no de una manera trivial. Todas estas restricciones del lenguaje de sombreado conducen a ejecuciones extremadamente rápidas.

```
1  shader_type canvas_item;
2  render_mode blend_premul_alpha;
3
4
5  uniform float _radius: hint_range(0.1 , 300.0) = 30.0;
6  uniform vec2 _mousePos;
7  uniform sampler2D _texture;
8  uniform vec4 _brushColor : hint_color;
9  uniform bool _paint;
10 uniform float _opacity : hint_range(0.0 , 1.0);
11 uniform float _smoothness: hint_range(0.0 , 5.0);
12 uniform sampler2D _activeLayerTexture;
13 uniform bool _lockAlpha = false;
14 uniform int _blendingMode : hint_range(0,1) = 0; //0 - normal 1 - erase
15 uniform float _brushFlow : hint_range(0,1) = 1.0;//This controls the alpha build up in the same brush stroke (without lifting the pen)
16
17
18 float alphaBlend(float base , float new)
19 {
20     return base * (1.0-new) + new;
21 }
22 vec4 getPixelFinalColor(vec2 fragCoord , vec2 position , vec4 activeLayerFragmentColor , vec4 previousPixelColor)
23 > { ...
24 }
25 uniform bool _xSymmetry = false;
26 uniform bool _ySymmetry = false;
27 uniform float _xSymmetryOffset:hint_range(0,1);
28 uniform float _ySymmetryOffset:hint_range(0,1);
29 void fragment()
30 {
31
32
33     if(_paint)
34     { ...
35     }
36     else
37     {
38         COLOR = vec4(0); //when we do not paint we just create an empty image.
39     }
40
41
42 }
```

Figura 37: Muestra del shader que utiliza la herramienta pincel

4.2 Persistencia de los datos

UNED Paint utiliza varias estructuras de datos para conseguir la persistencia de los proyectos y de la configuración de este. Las dos más importantes son:

UnedPaintProjectData (./data_structures/UnedPaintProjectData.cs)

- Contiene información necesaria para la reconstrucción de los proyectos UNED Paint como imágenes, capas, paleta de colores, resolución y otros.
- Estos datos se almacenan en el disco en formato binario con la extensión .upp .

ApplicationData (./data_structures/ApplicationData.cs)

- Contiene información necesaria para la configuración del editor UNED Paint.
- Estos datos se almacenan en el disco en formato binario con el nombre de app.dat .

4.3 Versionado

Para rastrear correctamente los builds, generamos en tres situaciones un **fichero de versión** que contiene el hash correspondiente al commit en el sistema Git:

1. Al hacer **commit** a los cambios actuales del proyecto con la ayuda de un *post-commit hook*.
2. Al hacer **checkout** de un commit con la ayuda de un *post-checkout hook*.
3. Al **recompilar** el proyecto en Visual Studio con la ayuda de un comando *batch*, ver la Figura 38.

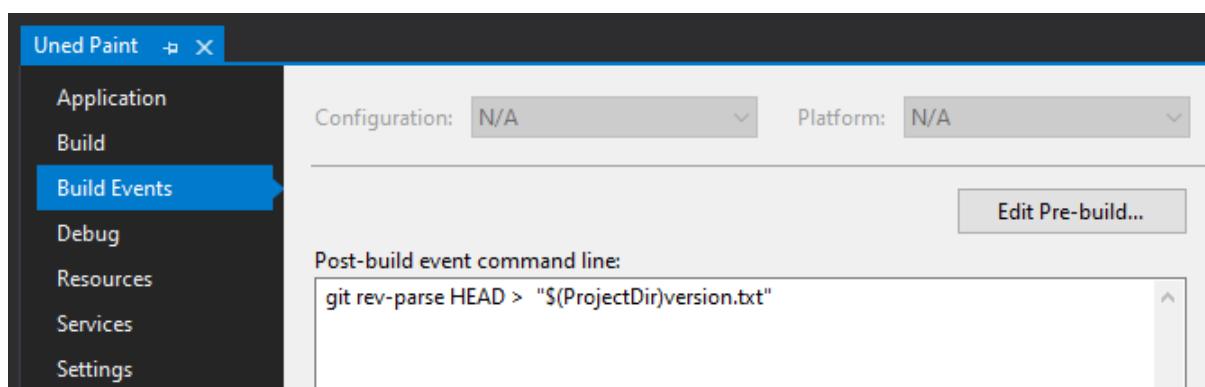


Figura 38: El script que genera el fichero de versión en Visual Studio 2019

Los *hooks*¹³ son ficheros bash, ver la Figura 39, ejecutados por el sistema de versionado Git.

```

1  #!/bin/sh
2  hash=$(git rev-parse HEAD)
3  pathToFile="./version.txt"
4  echo "$hash" > $pathToFile

```

Figura 39: Git hook que genera el fichero de versión

Al ejecutar, UNED Paint utiliza esta información. De esta forma si distribuimos el programa podemos consultar y rastrear rápidamente la versión del código fuente utilizando el hash correspondiente, ver la Figura 40.

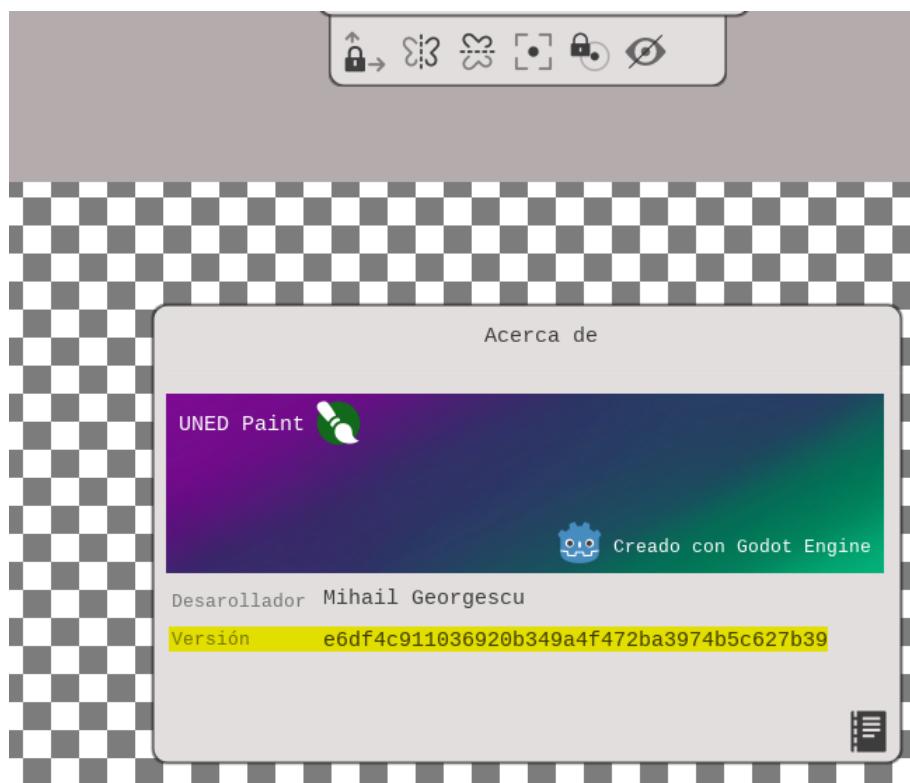


Figura 40: Ejemplo de consulta de versión dentro del editor

¹³Mas información sobre los git hooks <https://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks>

4.4 Internacionalización

Para implementar correctamente la internacionalización se ha utilizado el servidor de traducción de Godot, ver la arquitectura en la Figura 42. Este servidor es un **singleton** que se encarga de recuperar el string correspondiente a la llave indicada, del diccionario previamente creado. El diccionario es un documento .csv formateado de una manera específica, ver la Figura 43.

Desde el principio la aplicación se ha implementado teniendo en cuenta este tema. En vez de utilizar cadenas de caracteres literales, al cambiar el idioma, los consumidores (botones, etiquetas etc.) recuperan el string mediante el servidor de traducción facilitándole la llave, ver la Figura 41.

Actualmente UNED Paint esta traducido solamente al **español** e **inglés**.

```
string translation = TranslationServer.Translate("_loadingLabel");
```

Figura 41: Recuperando el string correspondiente a la clave _loadingLabel

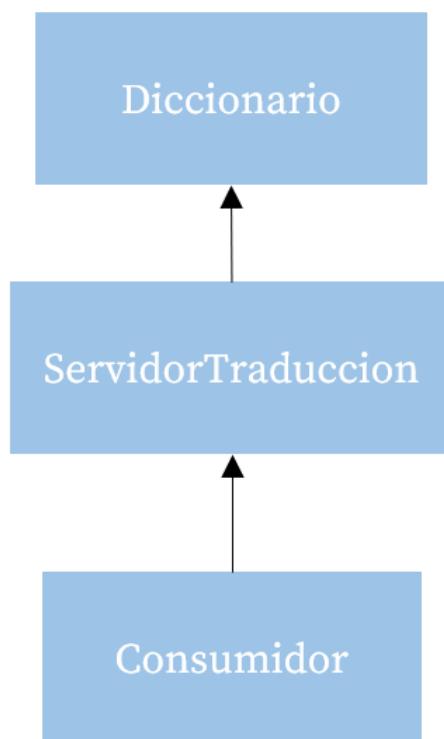


Figura 42: Arquitectura del sistema de traducción

	A	B	C
1	keys	en	es
2	resolutionLbl	Resolution	Resolución
3	rotationLbl	Rotation	Rotación
4	zoomLbl	Zoom	Zum
5	positionLbl	Position	Posición
6	en	English	English
7	es	Español	Español
8	languageSelectorTitle	Language	Idioma
9	interpolationToggleButton	Brush interpolation	Interpolación pincel
10	appSettingsTitleLbl	Settings	Preferencias
11	brushSettingsTitle	Brush settings	Configuración pincel
12	radiusSliderTitle	Radius	Radio
13	opacitySliderTitle	Opacity	Opacidad
14	flowSliderTitle	Flow	Flujo
15	smoothnessSliderTitle	Softness	Suavidad
16	spacingTitle	Spacing	Espaciado
17	newProjectButton	New Project	Proyecto nuevo
18	openProjectButton	Open file	Abrir fichero
19	resizeButton	Resize canvas	Cambiar el tamaño del lienzo
20	saveButton	Save	Guardar
21	saveAsButton	Save as	Guardar como
22	exportButton	Export	Exportar
23	settingsButton	Settings	Preferencias

Figura 43: Muestra del diccionario de traducción

4.5 Requisitos hardware y software para la ejecución

Mi intención fue que el editor por lo mínimo se ejecutase con cierta facilidad en la tablet que yo disponía. Las características más relevantes de la máquina son:

- Procesador: 7th Gen Intel® Core™ m3 2.6 GHz.
- RAM: 4 GB.
- Sistema Operativo: Windows 10.
- Tarjeta gráfica: Intel HD Graphics 615 con soporte para OpenGL 3.0.

El rendimiento del programa en esta máquina fue mi línea de base.

Requisitos software

- Sistema Operativo Windows.

Requisitos hardware

- Para que el editor ejecute correctamente la tarjeta gráfica necesita soporte para OpenGL 3.0.

4.6 Configuración del entorno de desarrollo

Para configurar correctamente el entorno de desarrollo se tienen que cumplir las indicaciones enumeradas a continuación.

1. Instalar el siguiente software:

- (a) Godot 3.2.2 MONO : https://downloads.tuxfamily.org/godotengine/3.2.2/mono/Godot_v3.2.2-stable_mono_win64.zip
 - (b) Visual Studio 2019 : <https://visualstudio.microsoft.com/es/vs/>
 - (c) Git SCM : <https://git-scm.com/>
 - (d) Mono 6.x+ for Windows : <https://download.mono-project.com/archive/6.10.0/windows-installer/mono-6.10.0.104-x64-0.msi>
- 2. Clonar** el repositorio UNED Paint : <https://github.com/georgescumihail-gh/UnedPaint> (A fecha de hoy, este repositorio es **privado**, necesitará una invitación como colaborador).
- 3. Copiar** el contenido encontrado en la carpeta *.hooks_backup* en la carpeta *./.git/hooks*.
- 4. Abrir** la solución *Uned Paint.sln* en Visual Studio y **Compilar** pulsando F6.

Para modificar y ejecutar las escenas de Godot tiene que abrir el archivo **project.godot** en el editor Godot, ver las Figuras 44 y 45, instalado anteriormente en el paso 1-a.

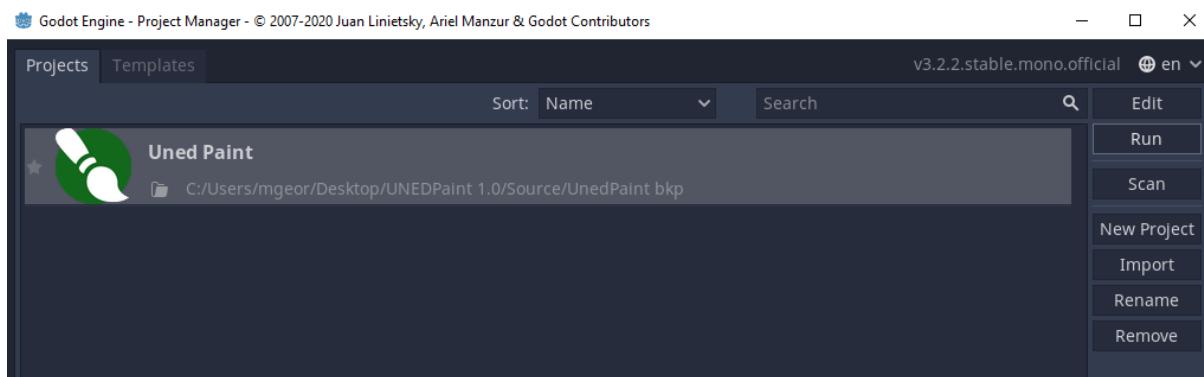


Figura 44: UNED Paint en el gestor de proyectos del editor Godot

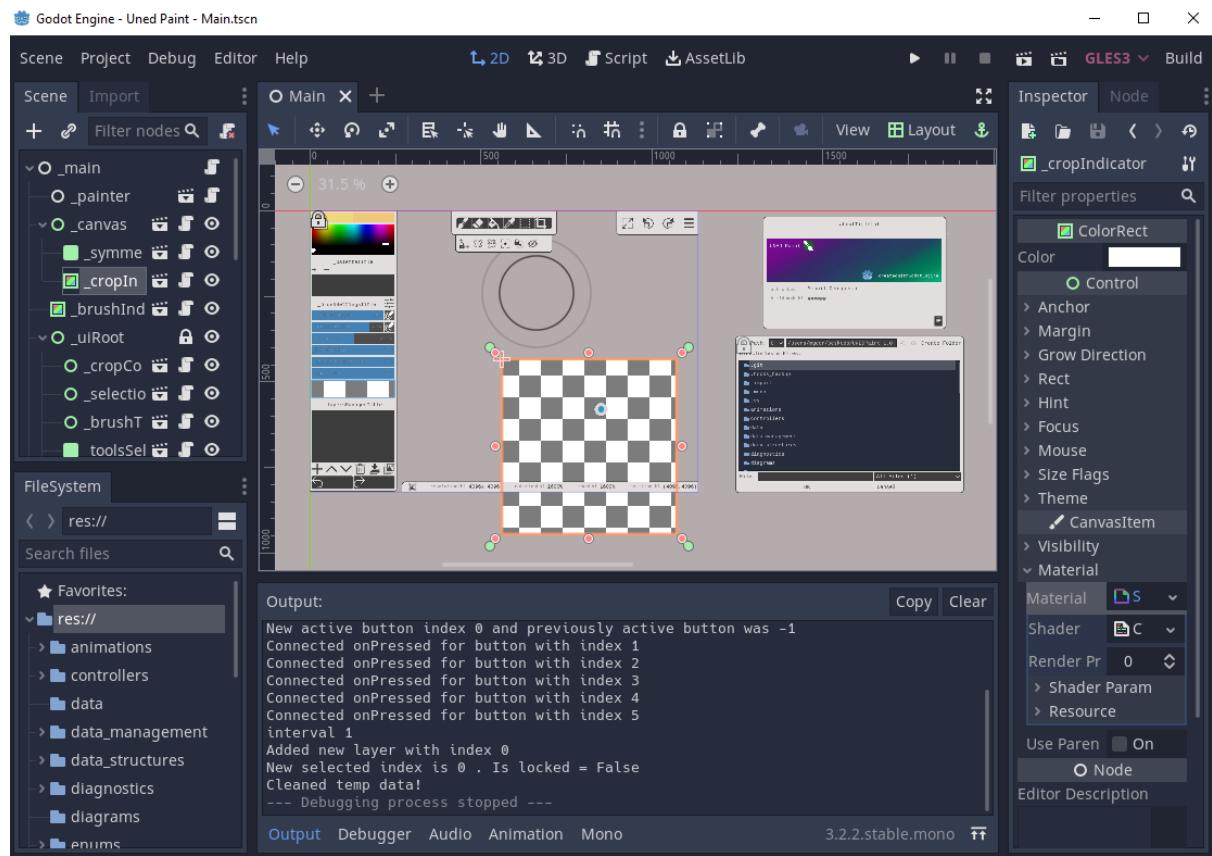


Figura 45: El proyecto UNED Paint abierto en el editor Godot

5 Pruebas

En el desarrollo de software es imprescindible hacer pruebas¹⁴ si el objetivo es conseguir un producto final de confianza. Crear y ejecutar pruebas de manera consistente reduce el riesgo de introducir errores en el programa. En un mundo ideal cada linea de código será comprobada por una prueba; en realidad ésto es bastante difícil de hacer. Para poder crear pruebas eficientemente es necesario escribir código de tal manera que sea comprobable. Esto se consigue siguiendo unos principios de desarrollo muy importantes como GRASP¹⁵ y SOLID¹⁶. En este proyecto se ha empleado una metodología de prototipo evolutivo. El proceso de desarrollo implicó un continuo aprendizaje de nuevas tecnologías y también del problema que se ha tenido que resolver, el problema de edición de imágenes. Con esto quiero decir que por la falta de experiencia con proyectos de relativamente gran escala y por la falta de tiempo, no siempre se respetaron los buenos principios de escritura de código, dificultando aún mas la creación de pruebas. Nuestro código necesitaría bastante mas refactorización para poder cubrir todas las funcionalidades con pruebas unitarias y pruebas de integración. Es un área de gran importancia que necesita sin duda una mejora.

5.1 Pruebas automatizadas

Actualmente no existe una manera fácil para realizar pruebas en Godot, por este motivo se ha tenido que implementar una herramienta experimental, ver la Figura 48, para conseguirlo de una forma bastante limitada.

El unit tester, llamado **GDTester**, se encuentra en `./diagnostics/unit_testing/GDTester.tscn`, ver la Figura 47.

Para que el tester detecte correctamente las pruebas, tenemos que cumplir los siguientes requisitos:

- Todas las pruebas se deben almacenar en la carpeta `./unit_tests/`
- Las clases de prueba necesitan heredar de **Node** y con el atributo **[GDUnitTest]** aplicado. Heredar de Node nos permite acceder a la funcionalidad del motor Godot.
- Los métodos que serán ejecutados tienen que ser **public** , con tipo de retorno **void** , **sin ningún parámetro** y aplicado el atributo **[GTestMethod]**

¹⁴https://es.wikipedia.org/wiki/Desarrollo_guiado_por_pruebas

¹⁵[https://en.wikipedia.org/wiki/GRASP_\(object-oriented_design\)](https://en.wikipedia.org/wiki/GRASP_(object-oriented_design))

¹⁶<https://es.wikipedia.org/wiki/SOLID>

Observa un ejemplo de prueba en la Figura 46.

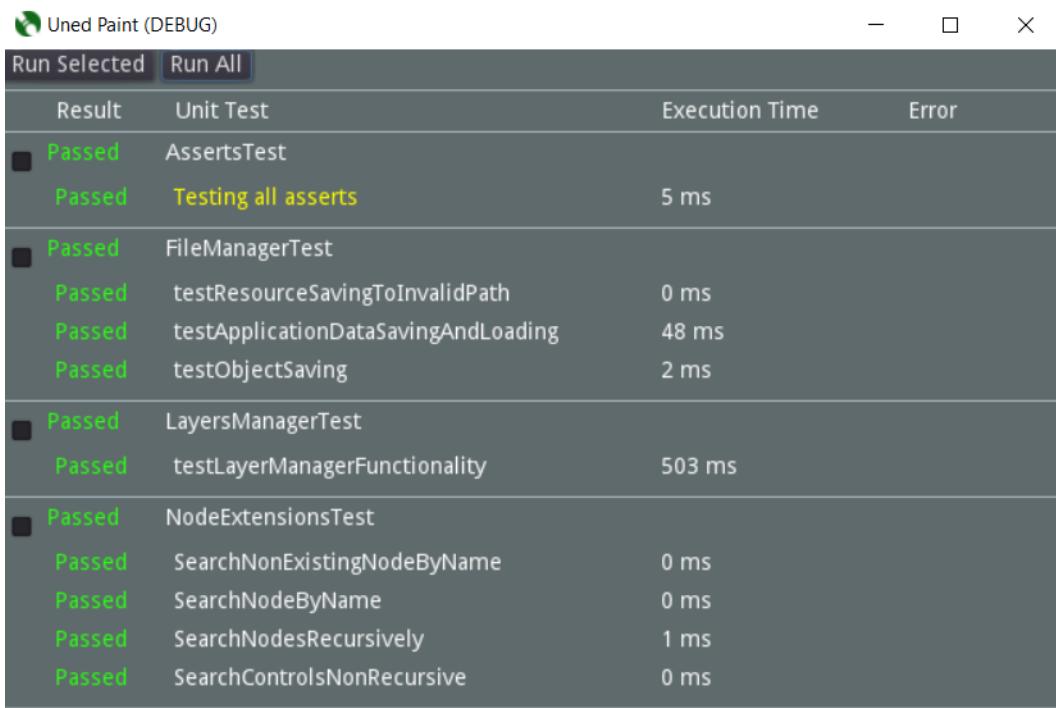
```
[GDUnitTest]
class NodeExtensionsTest : Node
{
    [GDTTestMethod]
    public void SearchNonExistingNodeByName()
    {
        Node node = this.GetNodeByName("Something");
        Assert.IsNull(node);
    }
}
```

Figura 46: Ejemplo de prueba en la herramienta GDTester



Figura 47: La carpeta del GDTester

Tal y como se puede observar en la Figura 48, la herramienta nos permite seleccionar las pruebas previamente creadas y ejecutarlas. Tras la finalización de la prueba, se muestra el estado de éxito de esta, el tiempo de ejecución y también información adicional si se da el caso.



The screenshot shows a window titled "Uned Paint (DEBUG)" with a toolbar at the top containing "Run Selected" and "Run All". The main area is a table with the following data:

Result	Unit Test	Execution Time	Error
Passed	AssertsTest		
Passed	Testing all asserts	5 ms	
Passed	FileManagerTest		
Passed	testResourceSavingToInvalidPath	0 ms	
Passed	testApplicationDataSavingAndLoading	48 ms	
Passed	testObjectSaving	2 ms	
Passed	LayersManagerTest		
Passed	testLayerManagerFunctionality	503 ms	
Passed	NodeExtensionsTest		
Passed	SearchNonExistingNodeByName	0 ms	
Passed	SearchNodeByName	0 ms	
Passed	SearchNodesRecursively	1 ms	
Passed	SearchControlsNonRecursive	0 ms	

Figura 48: Las pruebas de UNED Paint en la herramienta experimental GDTester

A continuación se explican brevemente las pruebas que aparecen en la Figura 48.

Prueba 1: AssertsTest

Comprueba los asertos del GDTester.

Prueba 2: FileManagerTest

Comprueba el correcto funcionamiento de la serialización y deserialización de los datos en UNED Paint.

Prueba 3: LayersManagerTest

Comprueba el correcto funcionamiento de las operaciones básicas como añadir, eliminar y desplazar capas de la clase LayersManager.

Prueba 4: NodeExtensionsTest

Comprueba las extensiones estáticas de la clase Node que se utilizan frecuentemente en todo el proyecto.

5.2 Pruebas manuales

Siendo una herramienta software muy visual y no disponer de ningún soporte nativo de pruebas por parte del motor Godot, fue bastante difícil comprobar el correcto funcionamiento de la mayoría de las funcionalidades de manera automática, por éste motivo muchas pruebas se realizaron de forma manual.

Las herramientas mas importantes de UNED Paint se implementaron utilizando sombreadores escritos en el lenguaje propio de Godot. En este momento desconozco la existencia de algún método de comprobación del correcto funcionamiento de éstos aparte del debugging visual.

Un ejemplo de prueba manual que se ha realizado es la prueba de mezclado de colores del sombreador de la herramienta pincel. La manera más fácil y rápida para comprobar el correcto funcionamiento de éste fue crear una imagen en UNED Paint, ver la Figura 49, y exportarla en otro editor de confianza, elegí Krita ver la Figura 50 en la parte superior de ésta podemos observar nuestra imagen creada en UNED Paint. En Krita se ha reproducido la misma imagen utilizando los mismos colores, ver la Figura 50 en la parte inferior de ésta. Posteriormente se muestraron las diferentes regiones de colores y se compararon los valores, ver la Figura 51, en la que se puede apreciar que éstos son iguales.

Es una manera de realizar pruebas muy primitiva y poco elegante pero en mi caso me ayudó confirmar el correcto funcionamiento del sombreador.

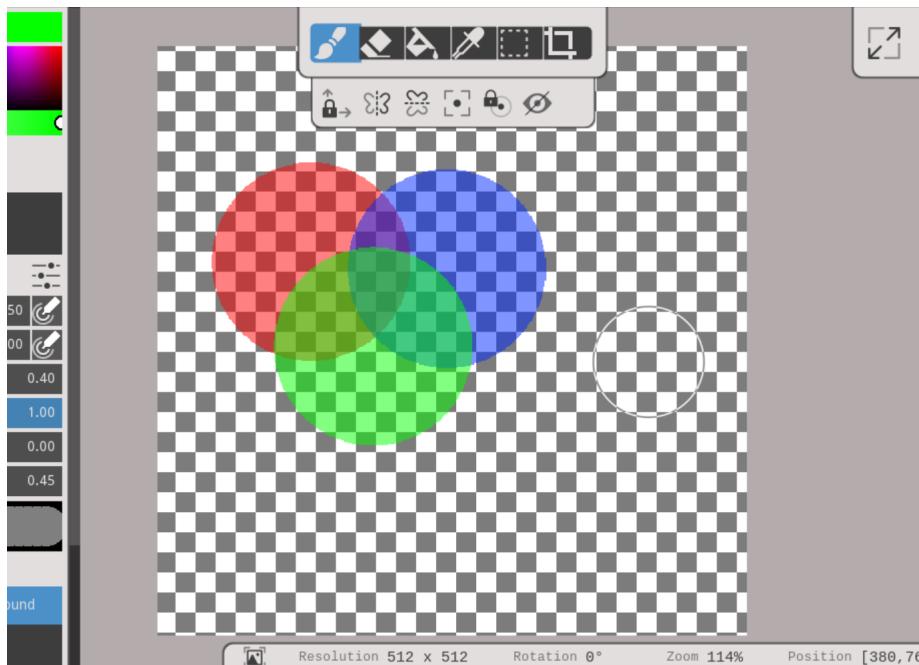


Figura 49: Imagen de prueba creada en UNED Paint

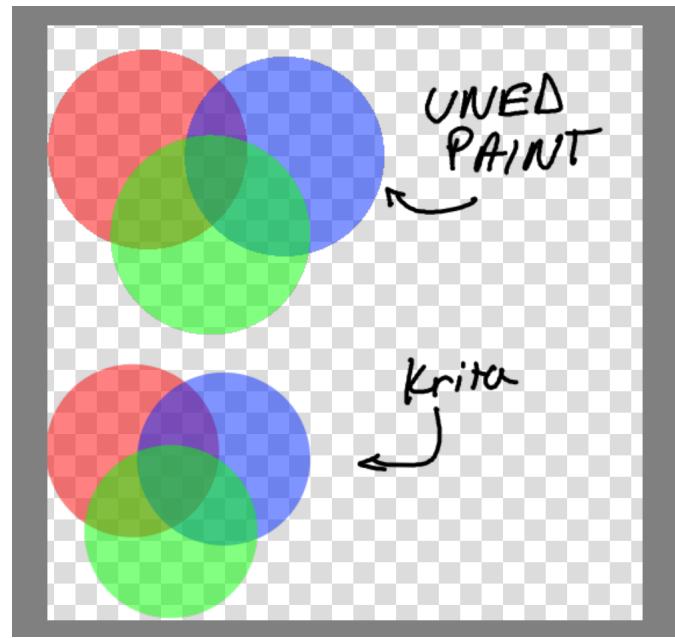


Figura 50: Imagen de prueba en el editor de imágenes Krita



Figura 51: Las diferentes regiones de colores y los valores RGBA correspondientes

6 Planificación y presupuesto

Para la organización de las actividades se empleó el método **Kanban**¹⁷ con la ayuda de la herramienta **Trello**¹⁸ y el sistema **Issues**¹⁹, ver la Figura 53, de GitHub.com.

Kanban es un sistema de gestión de actividades visual, ver la Figura 52. Kanban nos permite distribuir el trabajo en diferentes contenedores dependiendo del estado de la actividad. Se crearon 4 contenedores **TODO**(Actividades e ideas todavía no empezadas), **Work in progress** (Actividades en progreso), **Done** (Actividades completadas), **Fix**(Actividades que necesitan una segunda vista, problemas por solucionar). Resultó bastante difícil gestionar la gran cantidad de actividades generadas.

Poco a poco, a lo largo del proyecto, toda la gestión de las actividades se ha transferido al sistema **Issues** de GitHub.com por ser mucho más intuitivo, rápido y conveniente para mi flujo de trabajo. Las etiquetas y el complejo sistema de búsqueda facilitaron la organización del trabajo.

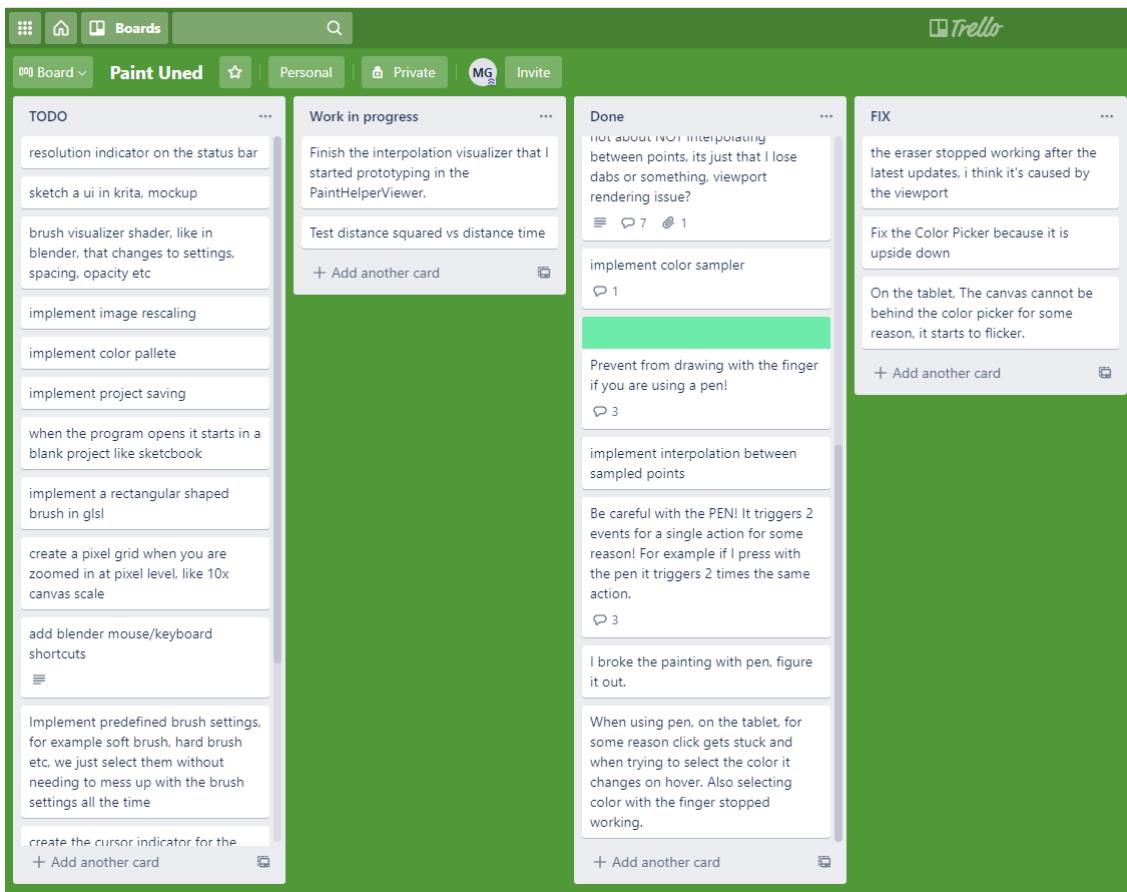


Figura 52: Ejemplo de organización de actividades Kanban en Trello

¹⁷<https://es.wikipedia.org/wiki/Kanban>

¹⁸<https://trello.com/es>

¹⁹<https://guides.github.com/features/issues/>

georgescumihail-gh / UnedPaint Private

<> Code Issues 2 Pull requests Actions Projects Security Insights Settings

Filters ▾ Clear current search query, filters, and sorts

0 Open 71 Closed

ⓘ Create an UI that controls the fill tool parameters Archived enhancement low priority
#103 by georgescumihail-gh was closed on Oct 20

ⓘ CanvasController input Implemented enhancement
#102 by georgescumihail-gh was closed on Oct 16

ⓘ Brush indicator color Implemented enhancement low priority
#101 by georgescumihail-gh was closed on Oct 15

Figura 53: Una muestra de las ”issues” de UNED Paint en GitHub.com

El tiempo estimado dedicado para el desarrollo de UNED Paint es de 950 horas, observar el diagrama de Gantt en la Figura 54.

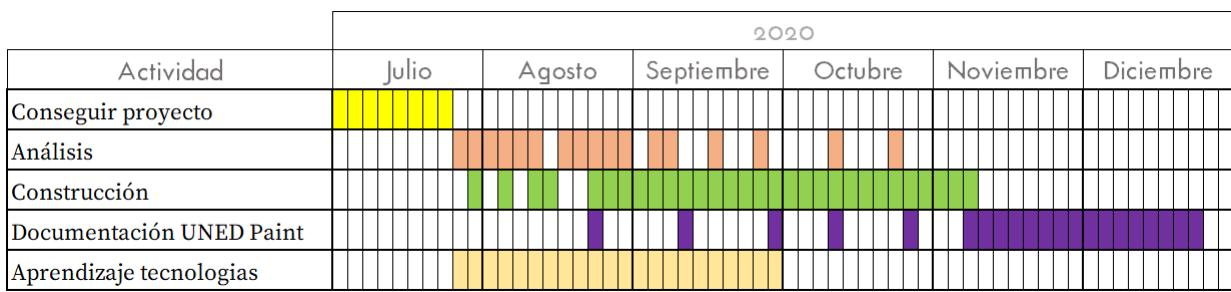


Figura 54: Diagrama Gantt a grosso modo de las actividades realizadas

6.1 Presupuesto

Para el calculo del presupuesto se utilizaron datos reales.

Según varias fuentes estadísticas de España e internacionales como *Glassdoor.es*, *Stackoverflow.com* o *Salary.com*, el salario medio para un trabajador en el puesto de *SOFTWARE DEVELOPER* podría ser de **4831 €**, ver la Figura 55. Sabemos que el trabajo se ha realizado en un periodo estimado de **5 meses** por una persona. El presupuesto del proyecto UNED Paint es $5 \text{ meses} \times 4\,831 \text{ €/mes} \times 1 \text{ desarrollador} = \mathbf{24\,155 \text{ €}}$.

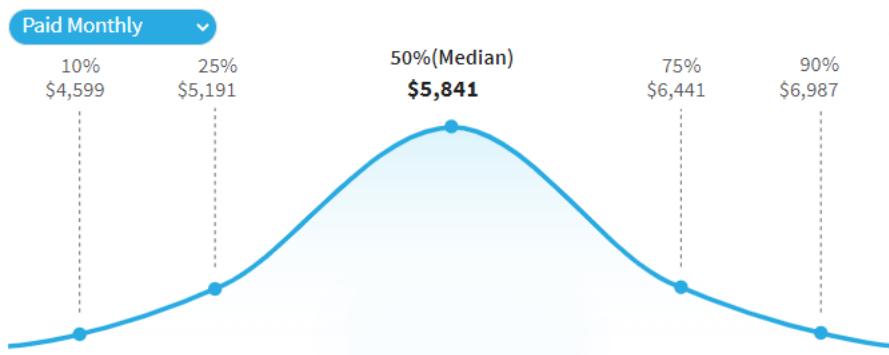


Figura 55: Salario medio mensual para un SOFTWARE DEVELOPER según Salary.com

6.2 Esfuerzo

Según *Visual Studio 2019 Metrics*, el proyecto UNED Paint consta de **10 000** líneas de código (10 KLOC). Utilizamos el metodo de estimación del esfuerzo COCOMO²⁰ en modo orgánico ($a = 2.4$ $b=1.05$).

Basic COCOMO

$$E = a(KLOC^b)$$

$$2.4(10^{1.05}) = 27 \text{ persona - mes}$$

Pienso que este resultado es muy exagerado ya que dependerá mucho del lenguaje de programación y otras variables pero por curiosidad se ha calculado.

²⁰<https://es.wikipedia.org/wiki/COCOMO>

7 Conclusiones y trabajos futuros

Para concluir quiero decir que los objetivos planteados se cumplieron con éxito en el plazo propuesto. Estoy bastante satisfecho con el resultado obtenido aunque algunas partes del proceso de desarrollo necesitarían una mejora como por ejemplo las pruebas. Este proyecto me ha permitido experimentar con diversas tecnologías y adquirir conocimiento que previamente no tenía, como por ejemplo trabajar con los lenguajes de sombreado algo que siempre me ha parecido interesante pero nunca tuve la oportunidad de aprender, por esto le agradezco mucho a **José Manuel Díaz Martínez** el director del proyecto y a los contribuidores que desarrollaron el motor de videojuegos Godot. En el futuro me gustaría profundizar más en el mundo de la informática gráfica y entender el funcionamiento de los API gráficos como DirectX, OpenGL o Vulkan.

7.1 Posibles cambios UNED Paint

A continuación presentaré los cambios más importantes que haría.

Interfaz de usuario

- Mejorar la interfaz de usuario puesto que tiene algunas inconsistencias.
- Modificar la barra de título de tal manera que sea consistente temáticamente con el resto del editor.

Funcionalidad

- Mejorar el sistema deshacer/rehacer para que registre otros tipos de cambios como por ejemplo el cambio de resolución y modificaciones del gestor de capas.
- Limitar la ejecución del editor a una única instancia. Implementar la detección de la instancia activa del editor mediante la comunicación interproceso.
- Implementar un sistema que permita el cambio de los atajos.
- Añadir modos de pintura como por ejemplo Multiplicar, Quemar, Luminosidad etc.

Otros

- Optimizar aún más el rendimiento de las diferentes herramientas, especialmente la herramienta pincel.

- Limpiar el código y refactorizar las API de los diferentes sistemas implementados
- Crear más pruebas unitarias y encontrar una forma más sencilla de hacerlos, actualmente esto es bastante difícil de conseguir en Godot sin crear una herramienta especializada.

7.2 Problemas conocidos

Ocasionalmente la barra de título deja de funcionar

Es un problema²¹ existente en la versión 3.2.2 de Godot. Esto se puede solucionar desenfocando y enfocando de nuevo la ventana del programa.

²¹<https://github.com/godotengine/godot/issues/33928>

Referencias y bibliografía

Editor de imágenes

- Editor de imágenes ráster: https://en.wikipedia.org/wiki/Raster_graphics_editor
- Edición de imágenes: https://en.wikipedia.org/wiki/Image_editing
- Imagen ráster: https://es.wikipedia.org/wiki/Imagen_de_mapa_de_bits
- Rasterización: <http://www.csc.villanova.edu/~mdamian/Past/csc8470sp15/notes/Rasterization.pdf>
- Editor de gráficos vectoriales : https://en.wikipedia.org/wiki/Vector_graphics_editor
- KRITA official website: <https://krita.org/es/>
- Sketchbook official website: <https://www.sketchbook.com/>

Sombreadores

- ¿Qué es un sombreador? (1): <https://es.wikipedia.org/wiki/Sombreador>
- ¿Qué es un sombreador? (2): <https://thebookofshaders.com/01/?lan=es>
- ¿Qué es un sombreador de pixel?: https://es.wikipedia.org/wiki/Sombreador_de_pixel
- El algoritmo Marching Ants: https://en.wikipedia.org/wiki/Marching_ants
- The Book of Shaders: <https://thebookofshaders.com/?lan=es>
- Shadertoy: <https://www.shadertoy.com/>
- Godot Shading Language: https://docs.godotengine.org/en/3.0/tutorials/shading/shading_language.html

OpenGL

- GLSL: <https://es.wikipedia.org/wiki/GLSL>

Documentación del lenguaje de programación C#

- Doc C#: <https://docs.microsoft.com/es-es/dotnet/csharp/>

Godot

- Godot página oficial: <https://godotengine.org/>
- La filosofía de diseño en Godot: https://docs.godotengine.org/es/stable/getting_started/step_by_step/godot_design_philosophy.html

Git

- Git Hooks: <https://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks>

GitHub

- GitHub official website: <https://github.com/>
- GitHub Issues: <https://guides.github.com/features/issues/>

Patrones de programación

- Composite pattern: https://en.wikipedia.org/wiki/Composite_pattern

Kanban

- ¿Qué es Kanban?: <https://es.wikipedia.org/wiki/Kanban>

El color

- Tono: [https://es.wikipedia.org/wiki/Tono_\(color\)](https://es.wikipedia.org/wiki/Tono_(color))
- HVS Colorspace: <https://www.ronja-tutorials.com/post/041-hsv-colorspace/>
- Premultiplicar el alpha: <https://microsoft.github.io/Win2D/html/PremultipliedAlpha.htm>
- Video youtube sobre los modos de "blending": https://www.youtube.com/watch?v=F7_kaTP7_W4

Definiciones interfaz de usuario “touch friendly”

- <https://www.instron.us/our-company/press-room/blog/2017/april/touch-friendly-bhui>
“... From a software standpoint, this means making items bigger and implementing gestures; sounds easy! What we've done to support the increased sizes for our inputs is to take a step back and try to simplify our workflows and use the available space on the screen in a smarter way... ”

Definiciones dibujo a mano alzada

- <https://glosarios.servidor-alicante.com/dibujo-tecnico/mano-alzada>
- <https://definicion.de/dibujo-a-mano-alzada/>

Metodologías de desarrollo

- Prototyping (1): https://en.wikipedia.org/wiki/Software_prototyping
- Prototyping (2): https://link.springer.com/referenceworkentry/10.1007%2F978-1-4020-8265-8_201039

Computación en tiempo real: https://en.wikipedia.org/wiki/Real-time_computing

Computación gráfica: [https://en.wikipedia.org/wiki/Computer_graphics_\(computer_science\)](https://en.wikipedia.org/wiki/Computer_graphics_(computer_science))

Apéndice A: Manual de usuario

En las siguientes secciones te explico brevemente las diferentes funcionalidades de la aplicación.

A.1. Pantalla principal

Al abrir la aplicación, te encuentras con la pantalla que te aparece en la Figura 56. El editor inicia un nuevo proyecto listo para trabajar en él.

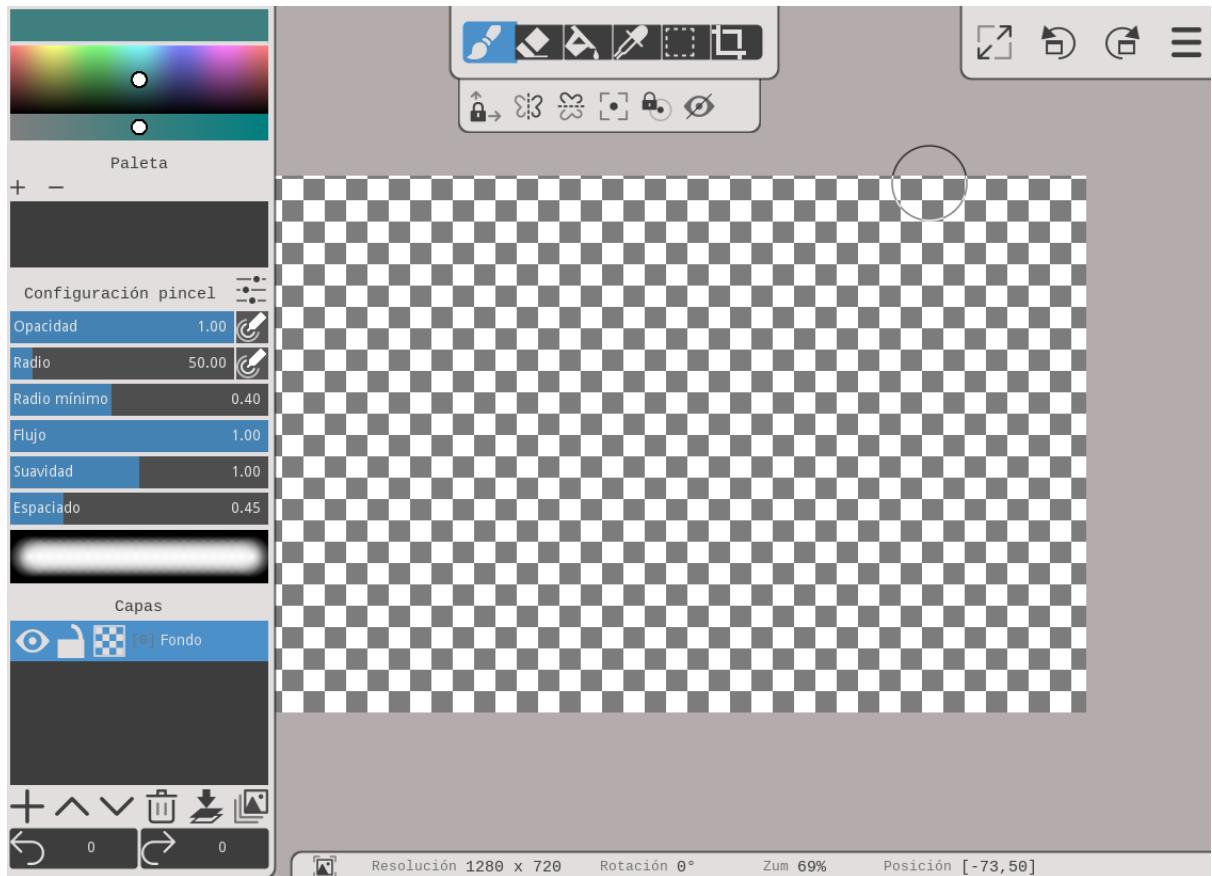


Figura 56: Pantalla principal de UNED Paint

A.2. El lienzo

En la Figura 57 observamos el lienzo. El patrón de tablero de ajedrez te indica que la imagen es completamente transparente, es una técnica común que utilizan los editores de imágenes . El lienzo puede ser transformado (trasladar, escalar, girar) utilizando los atajos (los encuentras en el panel de preferencias del editor) o la pantalla táctil.

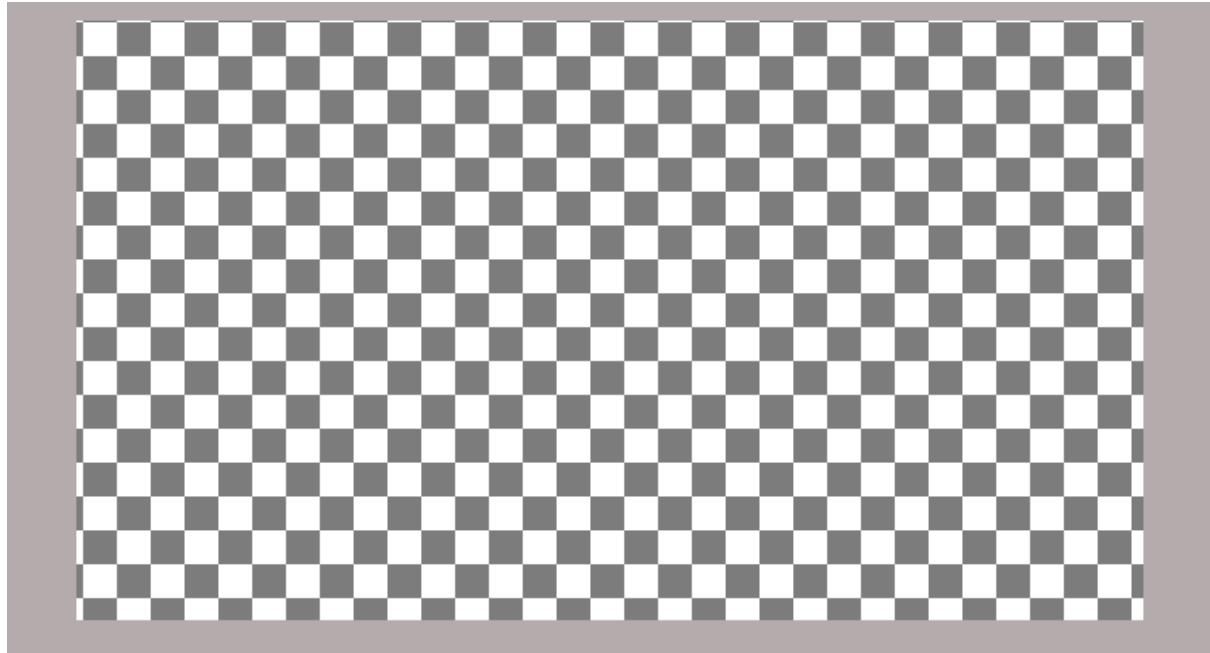


Figura 57: El lienzo

A.3. El panel de información del lienzo

El panel de información, ver Figura 58, te muestra información sobre el lienzo, como por ejemplo la resolución actual, la rotación, zum o posición.



Figura 58: El panel de información del lienzo

Es posible restablecer la transformación si pulsas el botón que aparece al principio de este.

A.4. El selector de herramientas

El selector de herramientas, ver Figura 59, te permite cambiar la herramienta que vas a utilizar.

El panel deslizante, el que aparece en la parte inferior de la Figura 59, modifica sus opciones en función de la herramienta elegida. Estas opciones controlan el comportamiento de la herramienta activa.

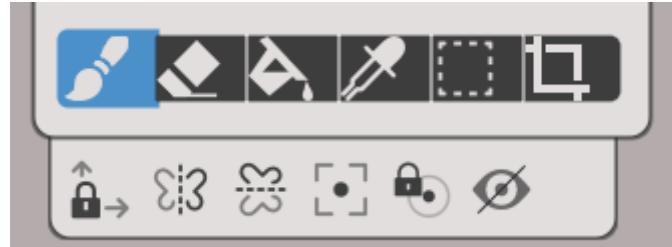


Figura 59: El selector de herramientas

A.5. Herramienta pincel

La herramienta te permite hacer dibujos a mano alzada. Presiona y arrastra sobre el lienzo para dibujar.

A.5.1. El panel de preferencias del pincel

El panel que se observa en la Figura 60, controla los diferentes parámetros del pincel.

El botón restablece los valores predefinidos.



Figura 60: El panel de preferencias del pincel

Opacidad

Controla la transparencia máxima del trazo; un valor de 0 hace el trazo completamente transparente y 1 totalmente opaco.



El botón de *opacidad controlada por presión*  se encuentra en la parte derecha del botón de opacidad.

Radio

Controla el radio máximo del pincel.



El botón de *radio controlado por presión*  se encuentra en la parte derecha del botón de radio.

Radio mínimo

Controla el radio mínimo del pincel, siendo un porcentaje del radio máximo. Este parámetro se utiliza para determinar el radio del pincel cuando el radio controlado por presión esta activado.

Flujo

El flujo controla el valor de transparencia de cada aplicación de color *sin levantar el lápiz*. Un valor de 1 aplica directamente el valor dado por la opacidad. Con un valor inferior a 1 necesitas *move el lápiz sobre la misma región varias veces* para acumular el color y alcanzar el valor máximo dado por la opacidad. En la Figura 61 puedes observar una tabla con diferentes valores de opacidad y flujo.

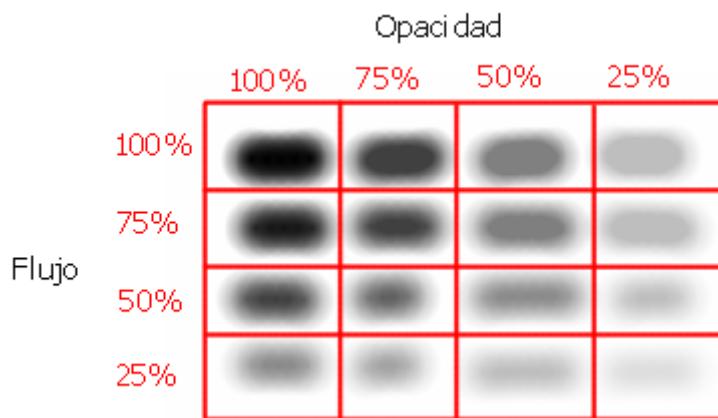


Figura 61: Ejemplos de trazos con diferentes valores de opacidad y flujo

En la Figura 62 podemos observar el mismo trazo en 3 instancias diferentes.

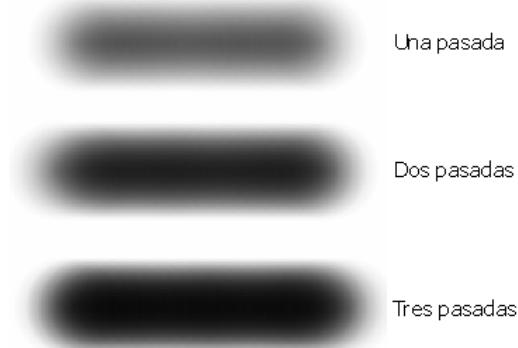


Figura 62: Ejemplo de trazo con opacidad 100% y flujo 25%

Suavidad

Controla la suavidad del trazo. Puedes observar el efecto de este parámetro en la Figura 63.

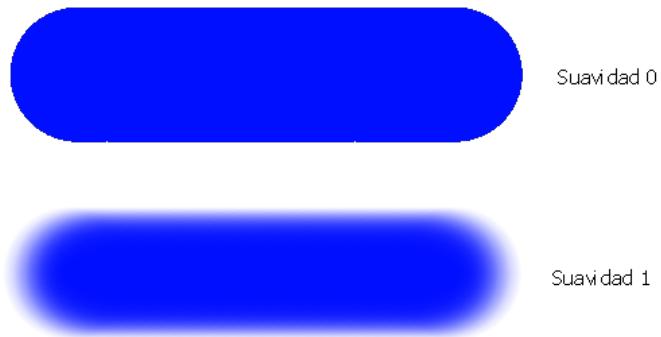


Figura 63: Ejemplos de trazos con diferentes valores de suavidad

Espaciado

Controla la distancia entre las aplicaciones de color del trazo. Puedes observar el efecto de este parámetro en la Figura 64.

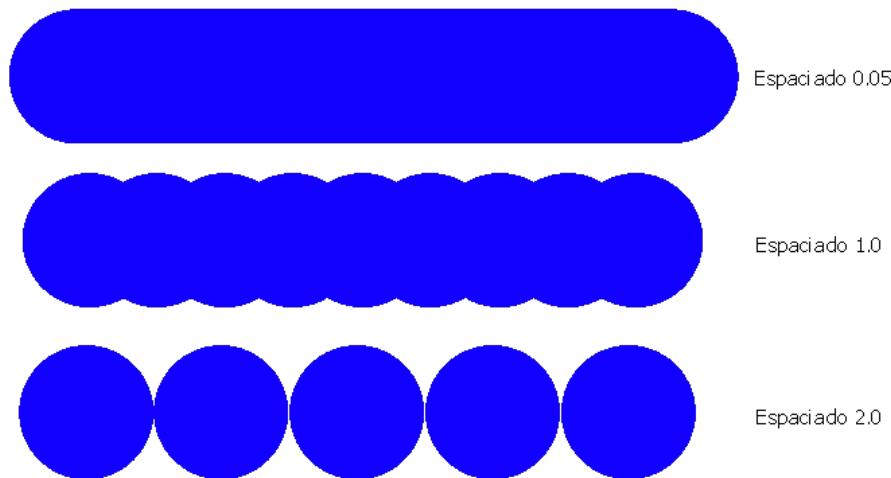


Figura 64: Ejemplos de trazos con diferentes valores de espaciado

A.6. El visualizador de trazos

El visualizador de trazos, ver la Figura 65, te ayuda previsualizar el trazo del pincel. Éste se actualiza con los cambios de las preferencias del pincel.



Figura 65: El visualizado de trazos

Si pulsas sobre él, realiza una prueba de velocidad del módulo que se encarga de “renderizar” el trazo. Los resultados de esta prueba te aparecen en el panel de notificaciones en la parte derecha de la ventana del editor, ver la Figura 66.

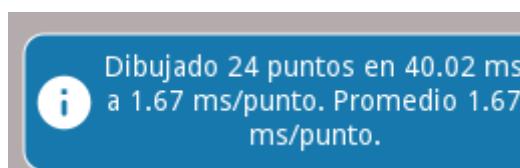


Figura 66: Resultado de la prueba de velocidad en el panel de notificaciones

A.7. El indicador circular del pincel

El color del indicador del pincel cambia dinámicamente con el fondo, ver la Figura 67. El circulo exterior tiene el radio igual al radio máximo alcanzable. El circulo interior tiene el radio igual al radio actual del pincel y cambia con la presión aplicada (solamente aplicable si utilizas un lápiz con un sensor de presión); este comportamiento lo activas/desactivas en el panel de preferencias del pincel.

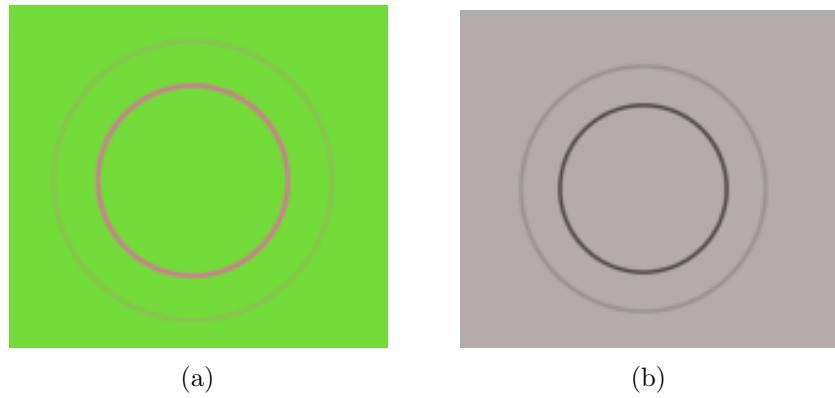


Figura 67: El indicador del pincel sobre diferentes fondos

A.8. Trazar líneas rectas

Para el trazado de líneas rectas tienes que mantener presionada la tecla que activa este modo (predefinido SHIFT) o el botón que se encuentra al principio del panel deslizante del selector de herramientas. El trazo puede ser vertical u horizontal y se basa en el movimiento inicial del ratón / lápiz . Ver un ejemplo en la Figura 68.

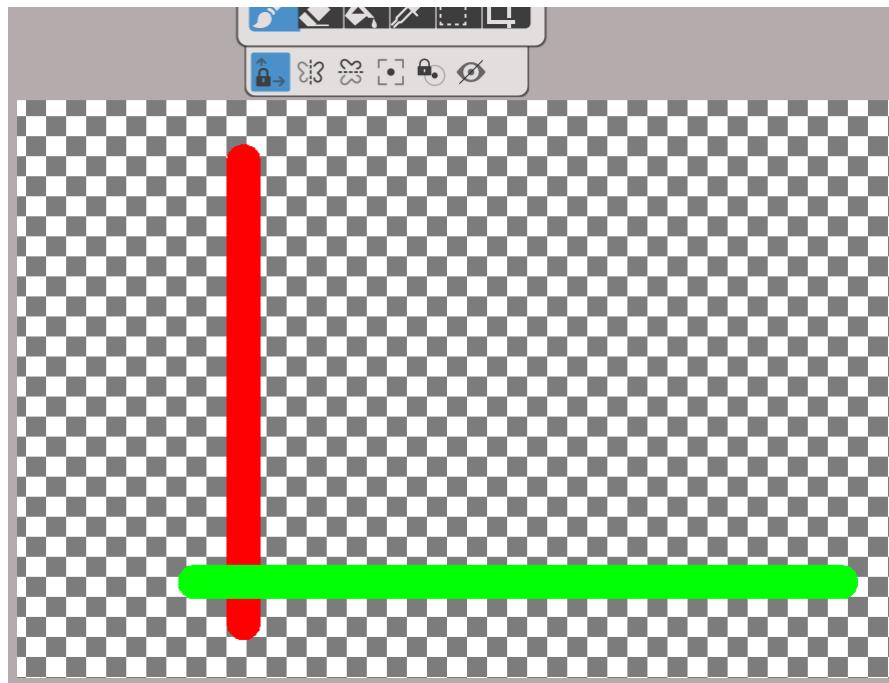


Figura 68: Ejemplo de trazado de lineas rectas

A.9. Pintura simétrica

Los botones que controlan la pintura simétrica, ver la Figura 69, los encuentras en el panel deslizante del selector de herramientas.

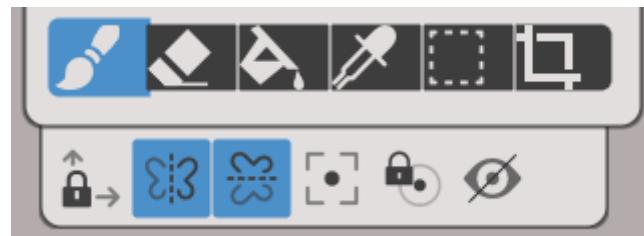


Figura 69: Los botones de activación de la pintura simétrica

El centro de simetría, ver la Figura 70, te aparece al activar algún modo de simetría y se puede mover en cualquier posición del lienzo. Observa un ejemplo de pintura simétrica en la Figura 71.



Figura 70: El centro de simetría

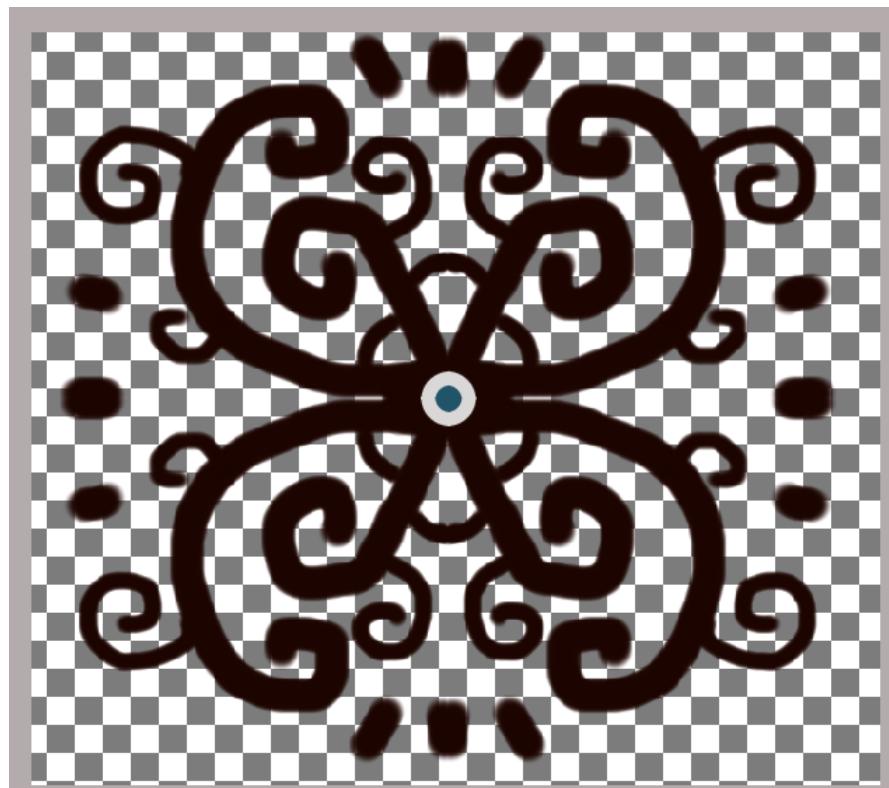


Figura 71: Ejemplo de dibujo simétrico vertical y horizontal

A.10. Herramienta de relleno

La herramienta de relleno cambia el color muestreado en la posición del lápiz y de todos los pixeles adyacentes similares a un nuevo color elegido. Por defecto, la herramienta muestrea *todas las capas visibles*. Ver los ejemplos que se muestran en las Figuras 72 y 73.

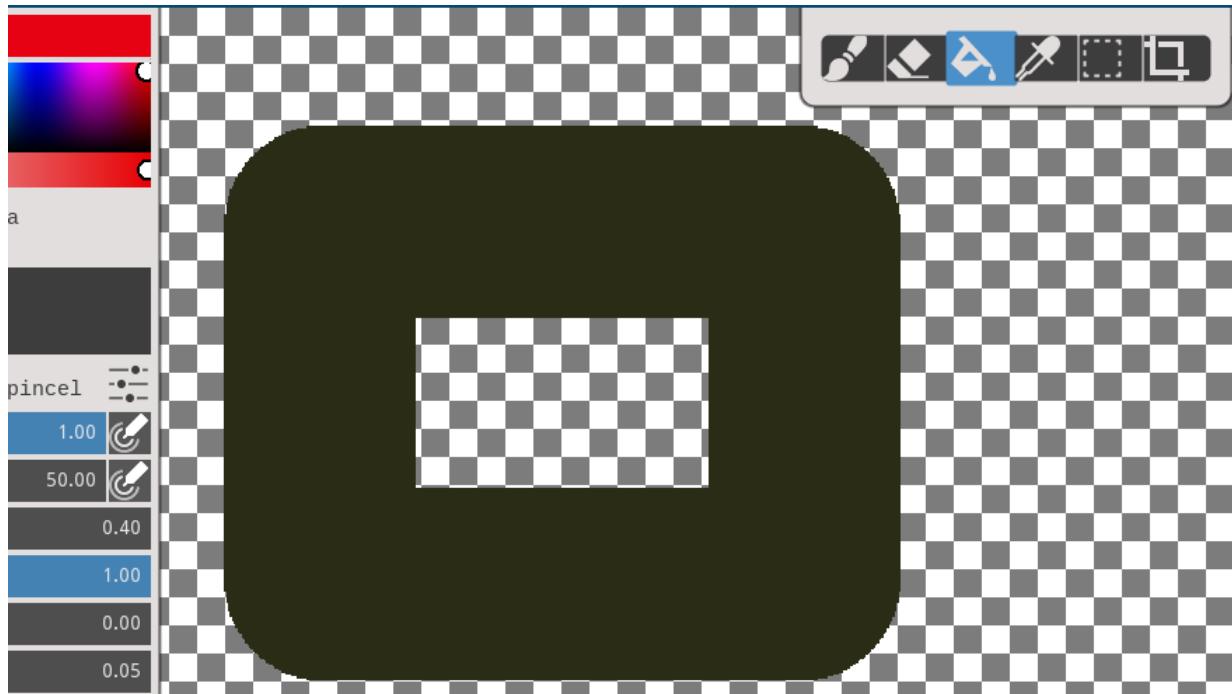


Figura 72: Imagen antes de utilizar la herramienta de relleno

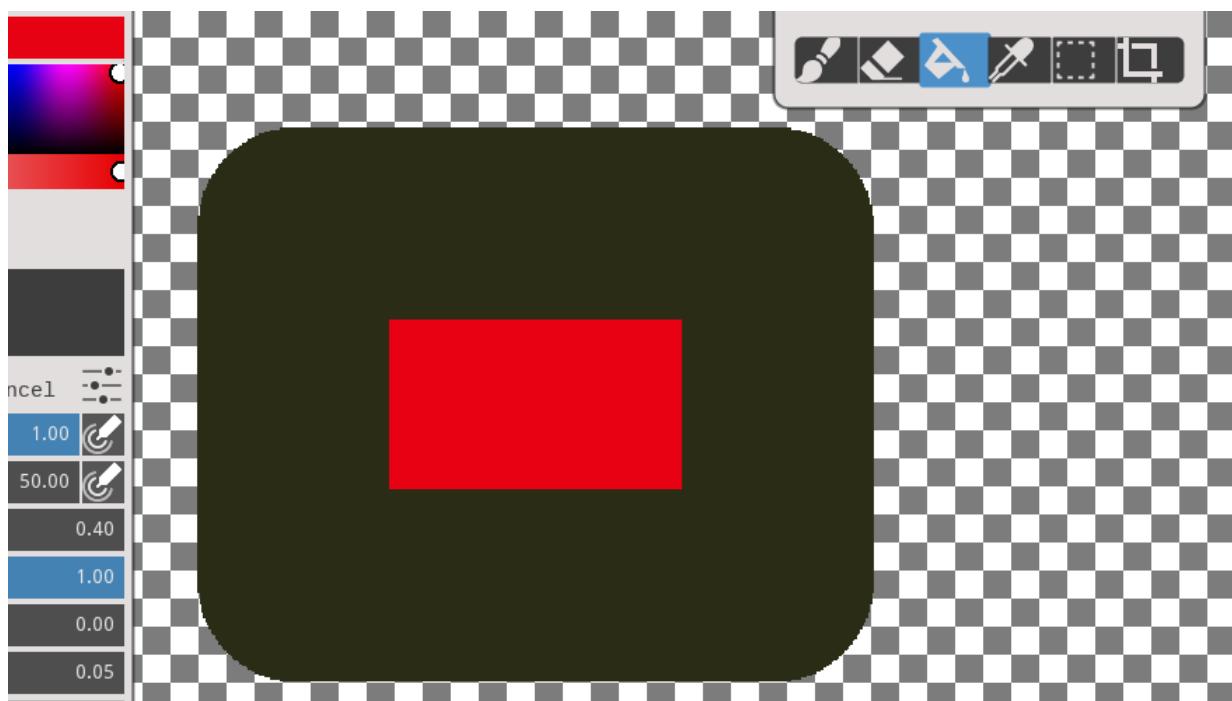


Figura 73: Imagen después de utilizar la herramienta de relleno

A.11. Herramienta de muestreo de color

La herramienta te permite muestrear el color del lienzo en la posición del ratón / lápiz. Pulsa en la posición deseada para la selección del color. Ver el ejemplo que se muestra en la Figura 74.

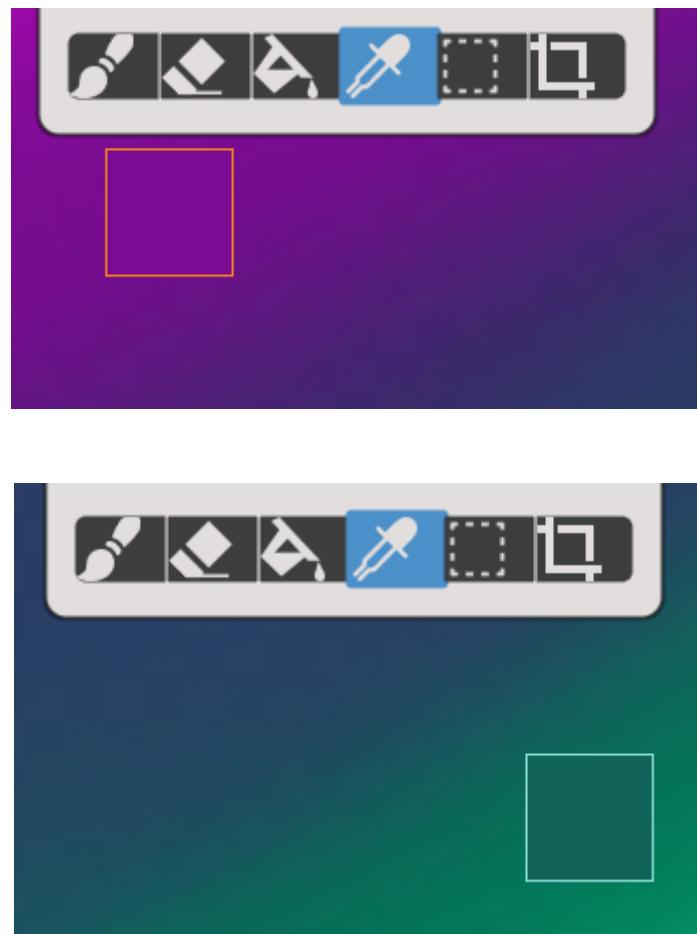


Figura 74: La herramienta de muestreo de color

A.12. Herramienta de selección

La herramienta de selección te permite seleccionar una región de la capa activa y desplazarla a una nueva posición en el lienzo. El proceso se muestra en la Figura 75 y 76 .

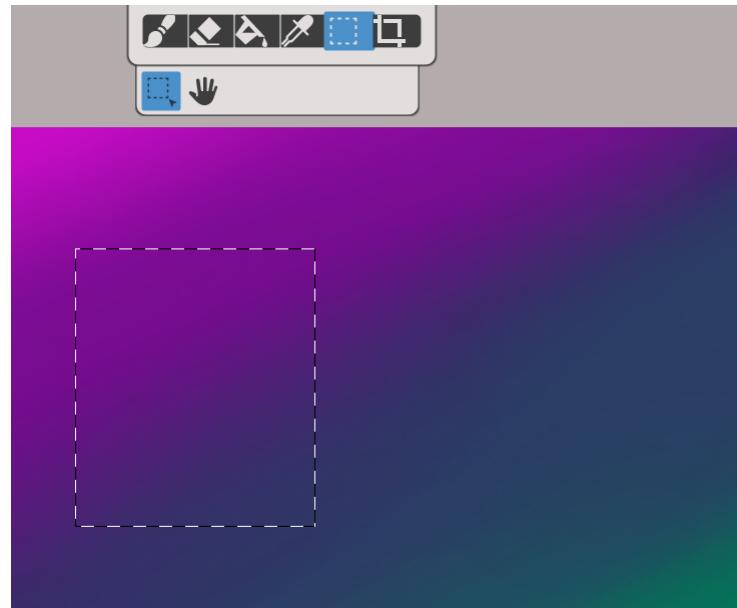


Figura 75: Dibujando el rectángulo de selección

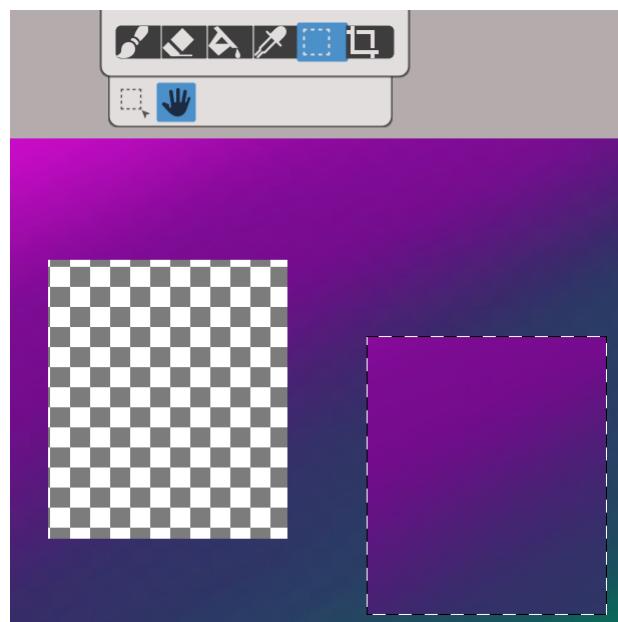


Figura 76: Moviendo la selección

A.13. Herramienta de recorte

La herramienta de recorte te permite seleccionar y recortar una región del lienzo, en un proceso muy sencillo de arrastrar sobre el lienzo para dibujar el rectángulo de recorte y aceptar pulsando el botón de aceptación. La nueva resolución del lienzo te aparece en el panel deslizante de la herramienta. El proceso completo se muestra en las Figuras 77, 78 y 79.

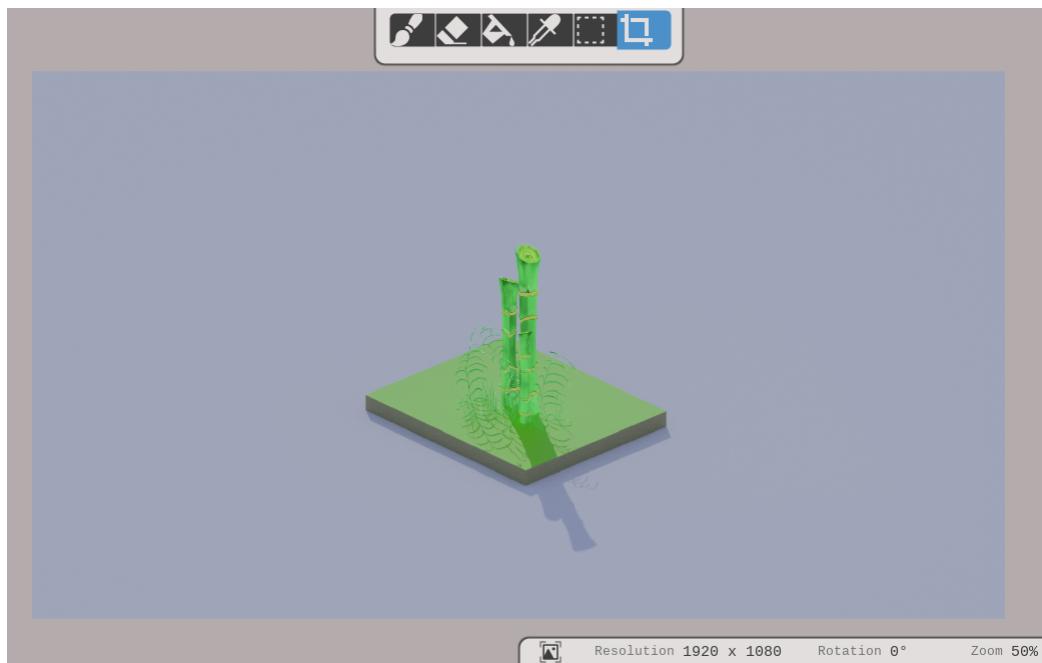


Figura 77: Imagen original antes de recortar

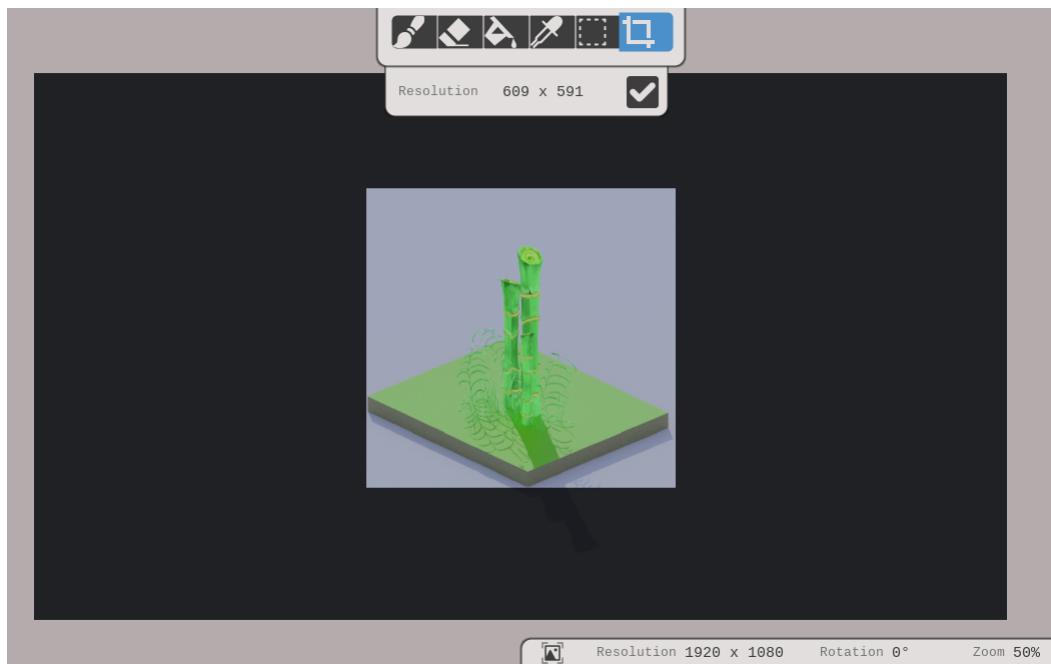


Figura 78: Dibujando la región de recorte

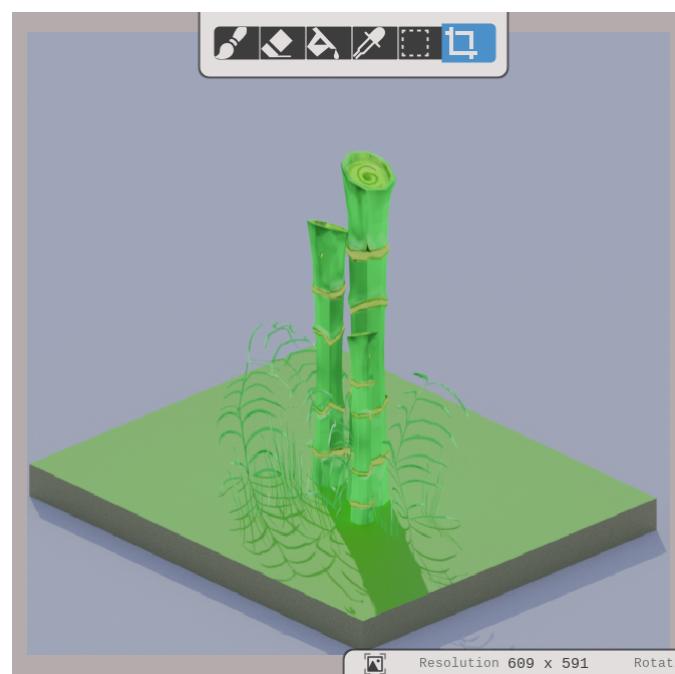


Figura 79: La nueva imagen recortada

A.14. El panel de control

El panel de control, ver la Figura 80, se encuentra en la parte superior derecha del editor. Realiza todas las operaciones para la gestión del proyecto y de la aplicación.

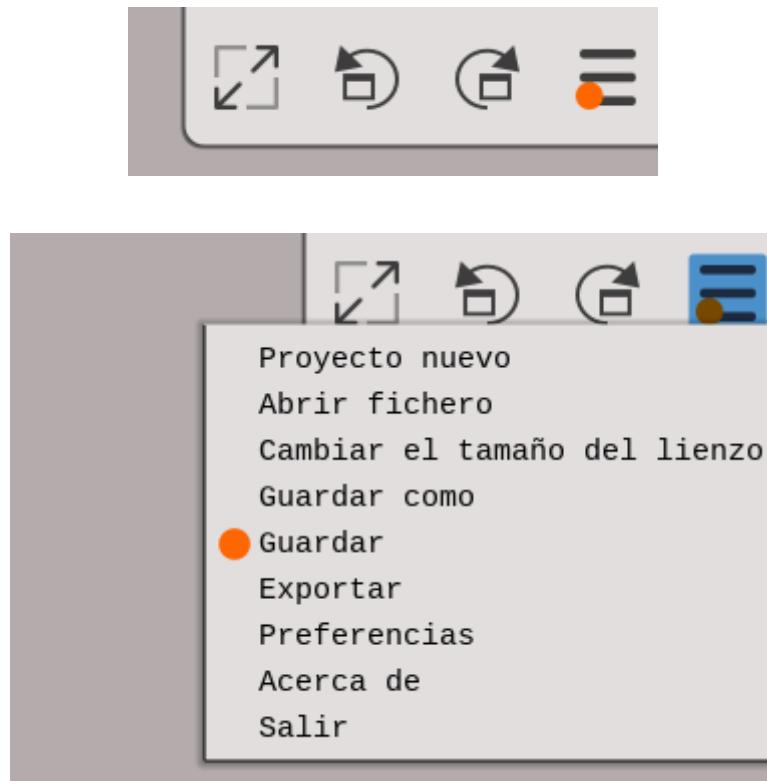


Figura 80: El panel de control

A.14.1. Crear un nuevo proyecto y redimensionar el tamaño del lienzo

Al elegir la operación “Proyecto nuevo” o “Cambiar el tamaño del lienzo” en las dos situaciones te aparece el panel de selección de la resolución, ver la Figura 81.

Puedes introducir manualmente una nueva resolución o elegir una predefinida. Si deseas mantener la proporción de la resolución al introducirla manualmente, tienes que activar el botón “cadena” ☰.



Figura 81: El panel de selección de la resolución

A.14.2. El indicador de cambios no guardados en el disco

Cuando en el proyecto actualmente abierto se detectan cambios no guardados en el disco, en la región del botón “hamburger” ☰ te aparece una bolita naranja, ver Figura 82, y desaparece al guardar el proyecto, ver Figura 83.



Figura 82: El panel de control con el indicador de cambios no guardados



Figura 83: El panel de control sin el indicador de cambios no guardados

Para evitar posibles pérdidas de datos, puedes activar la opción **guardar automáticamente** encontrada en el panel de preferencias, ver la Figura 84.

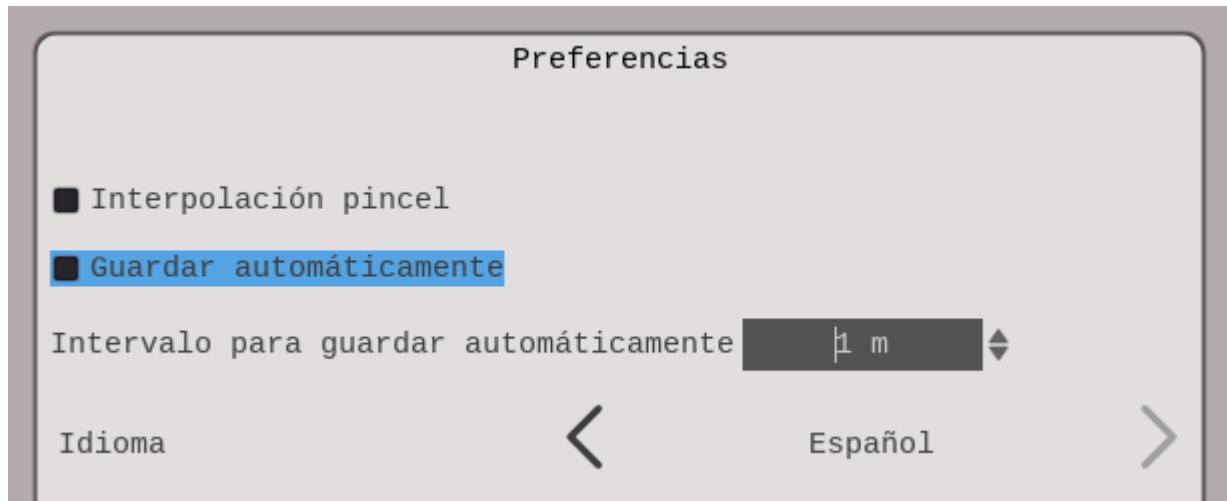


Figura 84: El panel de preferencias

A.15. Copia de seguridad

Al guardar el proyecto antes de sobrescribir el fichero .upp se creará también una copia de seguridad con el mismo nombre y con extensión .b kp. Puedes abrir esta copia de seguridad como cualquier otro proyecto .upp.

A.16. Rotación de la imagen

La rotación de la imagen se realiza mediante la pulsación de los botones, ver Figura 85, que se encuentran en la parte superior derecha en el panel de control. Se muestra un ejemplo en las Figuras 86 y 87.



Figura 85: Los botones de rotación de la imagen

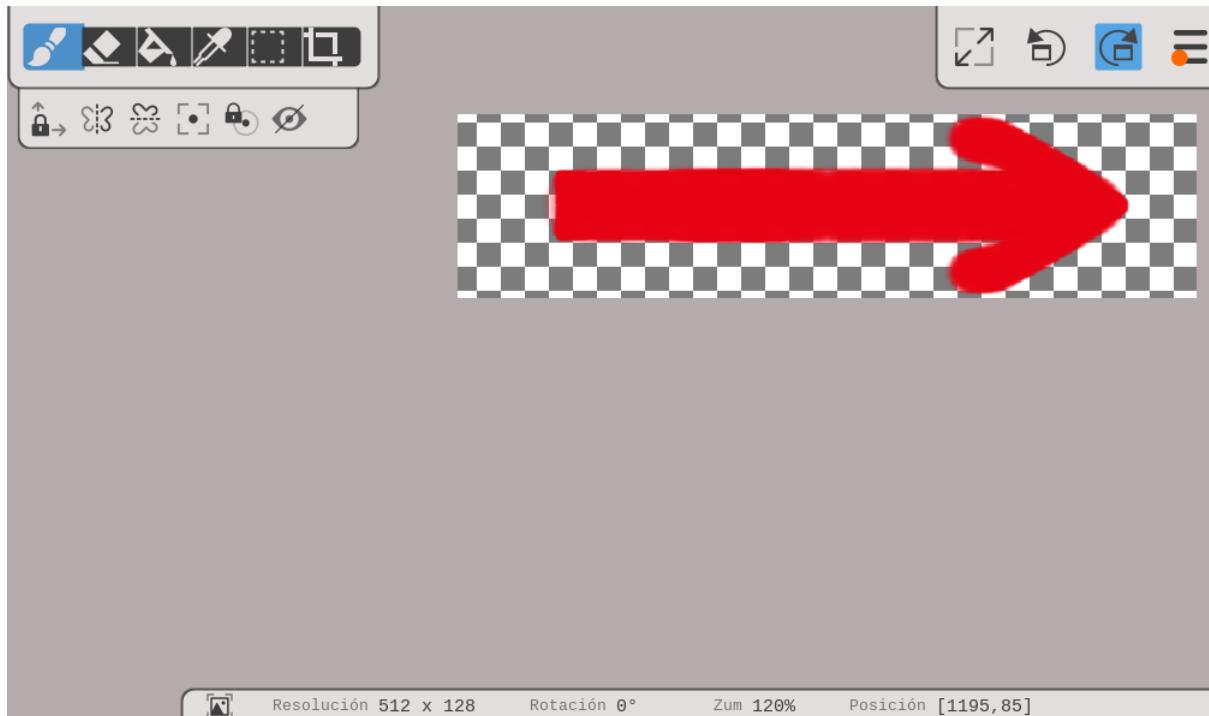


Figura 86: Imagen antes de realizar la rotación

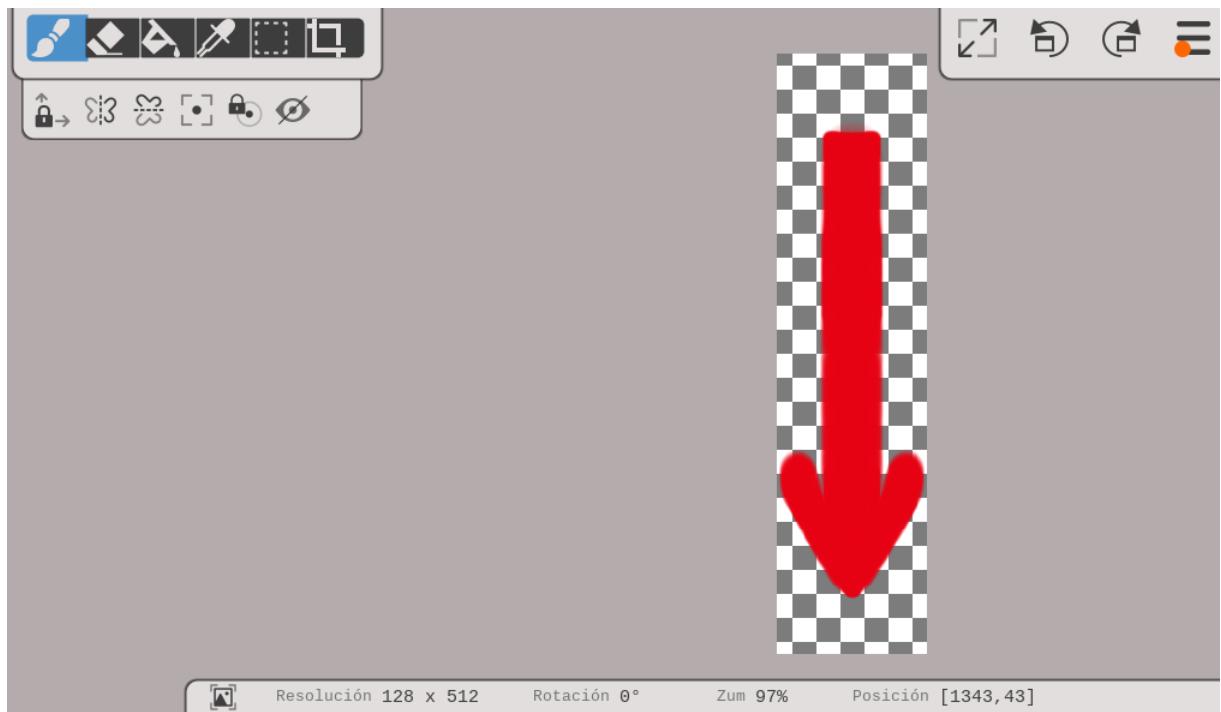


Figura 87: Imagen después de realizar la rotación a la derecha

A.17. La paleta de colores

La paleta de colores, ver la Figura 88, te permite guardar colores para su posterior uso.

Pulsa el botón que indica “+” si quieres añadir el color del selector a la paleta.

El botón “-” elimina el color elegido de la paleta.

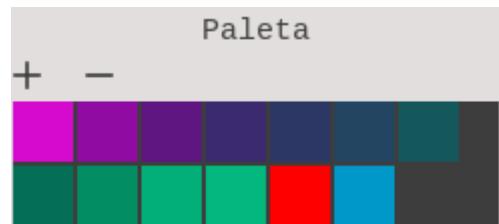


Figura 88: La paleta de colores

A.18. El selector de color

El selector de color, ver Figura 89, te permite cambiar el color que utiliza la herramienta activa arrastrando los indicadores circulares  sobre la posición del color que deseas.

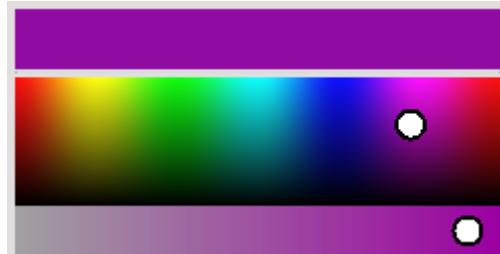


Figura 89: El selector de color

En la parte superior te aparece el color actual. Si arrastras algún indicador circular el color que te aparece en la parte superior se divide en dos partes para visualizar mejor la diferencia en los valores, ver Figura 90. En la parte izquierda tienes el color previo y en la parte derecha el nuevo color; al soltar, el indicador se rellena completamente por el nuevo color, ver Figura 91 .

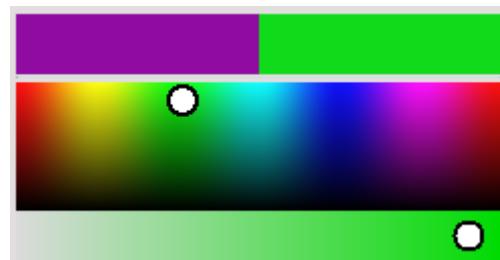


Figura 90: El selector de color, selectando un nuevo color

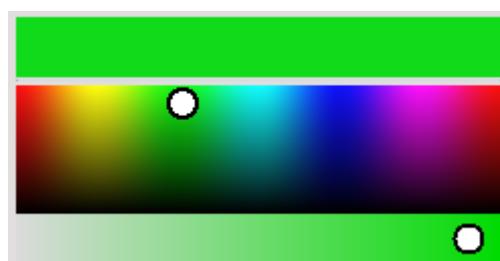


Figura 91: El selector de color, seleccionado un nuevo color

El indicador circular superior, controla el **tono** (HUE en inglés) en un movimiento horizontal y la **luminosidad** (VALUE, BRIGHTNESS, LUMINOSITY en inglés) en vertical.

Por último, el indicador circular que aparece en la parte inferior controla la **saturación** (SATURATION en inglés) y se puede mover solamente en horizontal, ver ejemplo de saturación baja en la Figura 92.

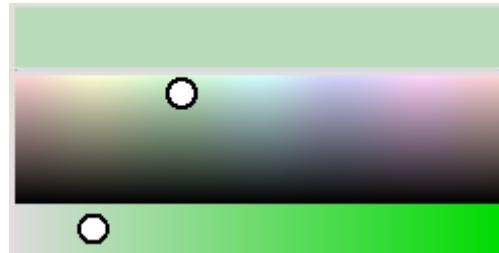


Figura 92: El selector de color, saturación baja

A.19. El gestor de capas

El gestor de capas te permite realizar diferentes operaciones sobre las capas:

- Añadir nueva capa
- Mover hacia arriba/hacia abajo la capa seleccionada
- Eliminar
- Fusionar hacia abajo la capa seleccionada
- Duplicar
- Ocultar
- Bloquear (al activar, la capa no se puede modificar, eliminar o mover).
- Preservar la transparencia (tal y como se ha implementado en la versión actual el algoritmo preserva solamente la transparencia completa) .
- Renombrar (pulsando dos veces seguido sobre el nombre de la capa).

Las capas se dibujan desde arriba hacia abajo. Puedes observar esto en las Figuras 93 y 94.

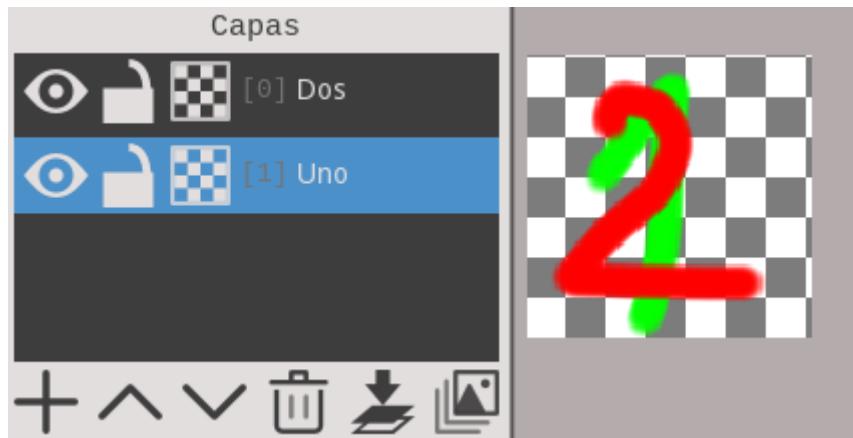


Figura 93: La capa con el nombre "Dos", dibujada por encima del resto de las capas

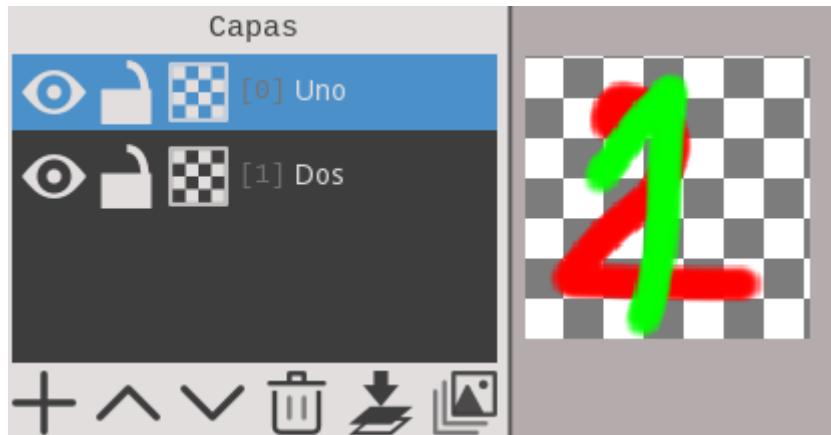


Figura 94: La capa con el nombre "Uno", dibujada por encima del resto de las capas

A.20. Deshacer yrehacer

Los botones deshacer yrehacer, ver Figura 95, se encuentran en la parte inferior izquierda del editor. En el interior de los botones se muestra el n mero de cambios que puedes deshacer / rehacer .



Figura 95: Los botones deshacer/rehacer

Nota: En la versi n actual se registran solamente los cambios de imagen del lienzo. No puedes deshacer los cambios de resoluci n o las modificaciones del gestor de capas.

A.21. El panel de preferencias

En el panel de la Figura 96 encuentras todas las preferencias y los atajos del editor.



Figura 96: El panel de preferencias

A.21.1. Interpolación pincel

Movimientos rápidos del ratón producen trazos interrumpidos . Al activar la interpolación el algoritmo intentará llenar los huecos. Ver los efectos de la interpolación en las Figuras 97 y 98.

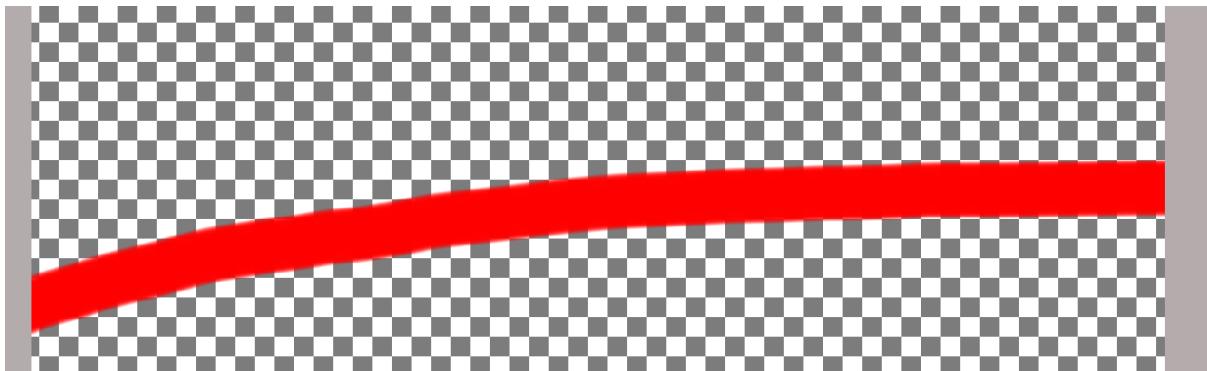


Figura 97: Trazo con interpolación

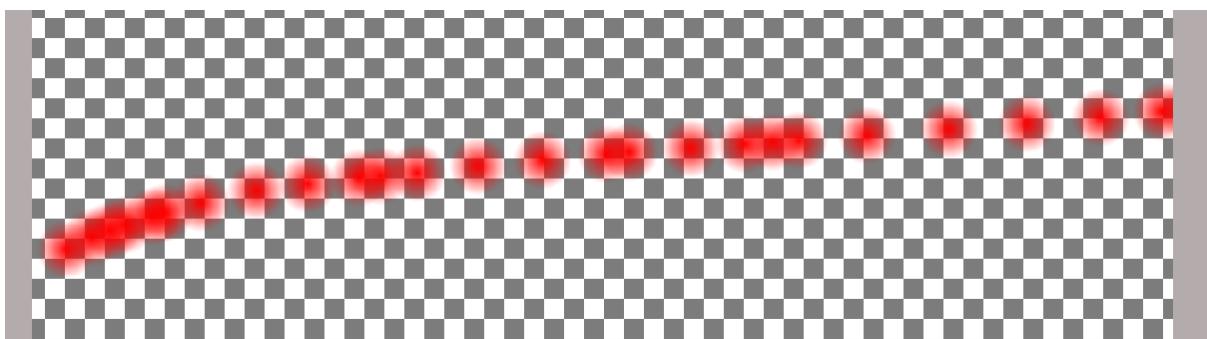


Figura 98: Trazo sin interpolación

A.21.2. Idiomas

La versión actual está traducida solo al **español** y al **inglés**. Puedes cambiar el idioma en el panel de preferencias del editor.

A.22. Cargar imágenes y proyectos

Arrastra y suelta en la ventana ficheros **.upp** (proyectos UNED Paint) para cargarlos. Si tienes cambios no guardados en el disco, el editor necesita una confirmación, ver Figura 99.

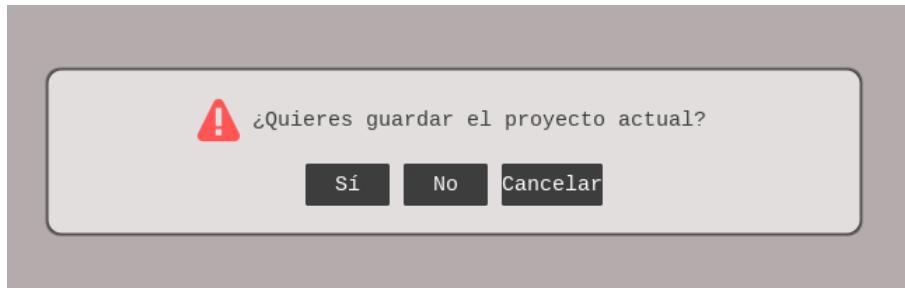


Figura 99: Ventana de confirmación de la operación

Arrastra y suelta en la ventana imágenes compatibles con el editor (.jpg, .png, .svg, .bmp). Te aparecerá un panel, ver Figura 100, con las siguientes opciones:

- **Escalar la imagen al tamaño del lienzo** (produce deformaciones si las resoluciones del lienzo y de la imagen no coinciden, ver Figura 101)
- **Escalar el lienzo al tamaño de la imagen** (ver Figura 102)
- **Sin cambios** (la nueva imagen se coloca en la parte superior izquierda del lienzo, si es menor que el tamaño del lienzo rellena el resto del espacio con píxeles transparentes; si es mayor que el tamaño del lienzo recorta la imagen, ver Figura 103)



Figura 100: Opciones de importación de una imagen

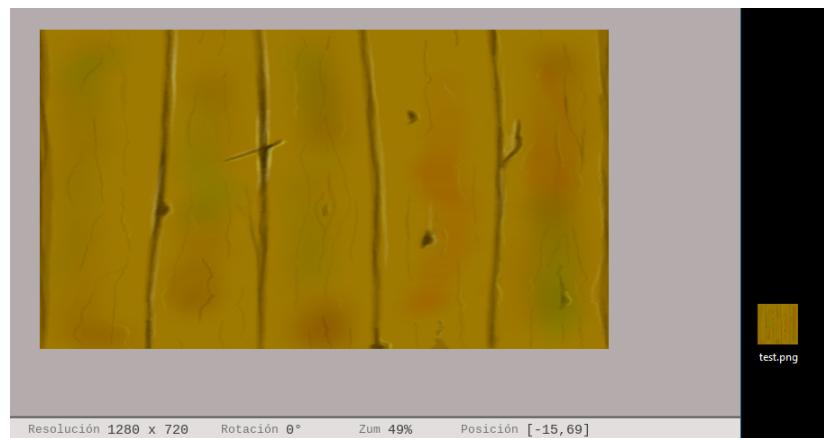


Figura 101: Opción Escalar la imagen al tamaño del lienzo

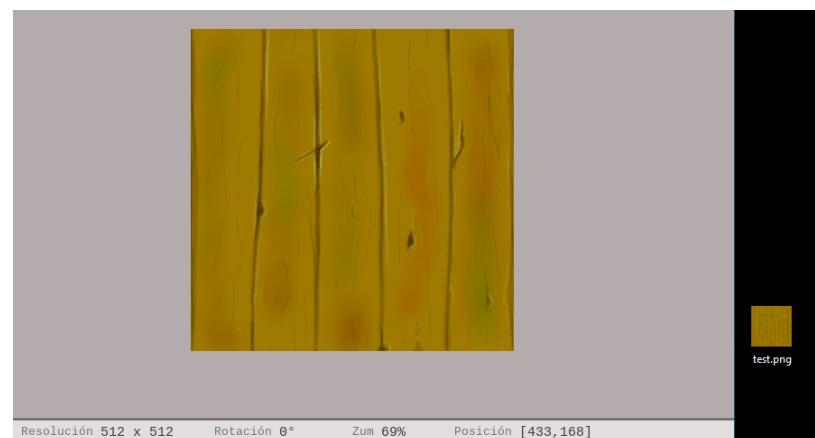


Figura 102: Opción Escalar el lienzo al tamaño de la imagen

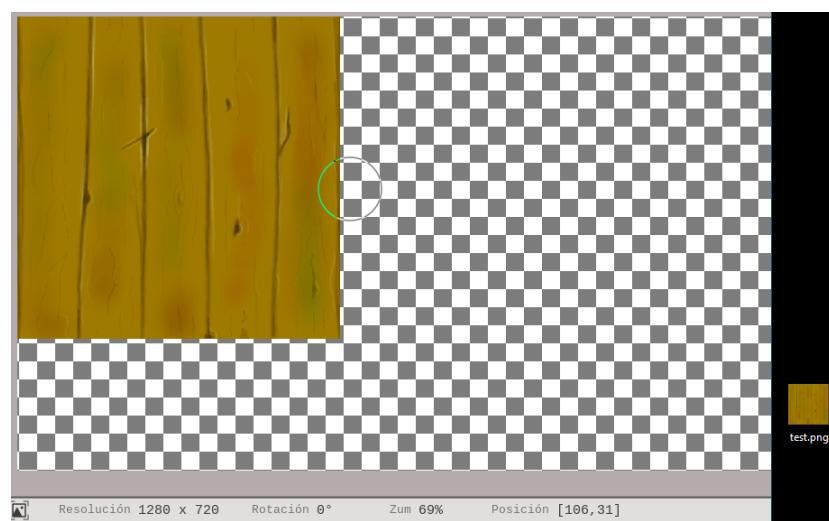


Figura 103: Opción Sin cambios

Al elegir cualquiera de las tres opciones , el editor añade una nueva capa con la imagen importada.

Todas las operaciones explicadas anteriormente se pueden realizar desde el panel de control seleccionando la operación “**Abrir fichero**”.

A.23. El panel de notificaciones

El panel de notificaciones es un panel deslizante que aparece en la parte derecha de la ventana tal y como se muestra en la Figura 104.

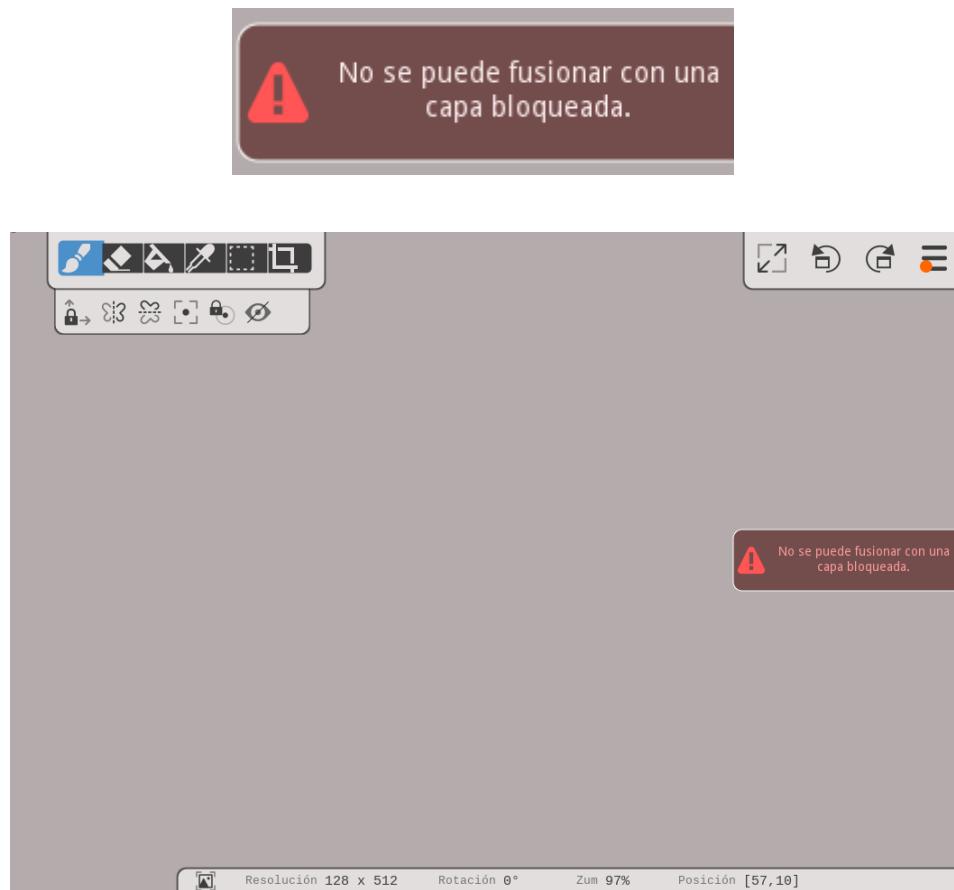


Figura 104: El panel de notificaciones

El editor utiliza este sistema para señalar diferentes eventos.

Pulsa el panel para ocultar el mensaje o espera varios segundos.

Version: March 23, 2021