

Disney · PIXAR
**INTENSA
MENTE**
EN
 **UTECH**





Proyecto Final del Curso
Título: Intensamente en UTEC

Grupo 4:
Irán Alvarez
Julisa Lapa
Adrián Sandoval
Libra Vento

Curso: (CS4016) Computación Gráfica
Profesor: Eric Biagioli

Índice

1	Introducción	2
2	Objetivos	2
3	Marco Teórico	2
3.1	Poisson Image Blending	3
3.2	Color Transfer	4
3.3	Alpha Blending	6
4	Desarrollo	6
4.1	Guion	6
4.2	Animación 3D	7
4.3	Implementación del blending	11
4.3.1	Selección de Imágenes	11
4.3.2	Masking	12
4.3.3	Aplicación de las técnicas	12
4.3.4	Environment	13
4.3.5	Implementación en código	14
4.4	Edición del Cortometraje	15
4.4.1	Recolección de Recursos Adicionales	15
4.4.2	Edición del Vídeo	15
5	Conclusión	16
5.1	Resultados y Reflexión	16
5.2	Limitaciones y Futuras Mejoras	16

1. Introducción

¿Qué pasaría si pudiéramos ver nuestras emociones? ¿Y si cobraran vida justo cuando más las necesitamos? En la vida universitaria, cada desafío, logro o frustración despierta un torbellino emocional. Alegría, miedo, enojo, tristeza y ansiedad no solo nos acompañan: son protagonistas invisibles que moldean nuestras decisiones y experiencias. Con este proyecto, decidimos darles un rostro.

Inspirados por la película *Intensamente*, creamos un cortometraje que mezcla arte, tecnología y sentimiento. Aplicando técnicas de computación gráfica como *Poisson Blending*, *Alpha Blending* y *Motion Transfer*, llevamos a los personajes de la película al entorno real de la universidad UTEC.

No solo fue un ejercicio técnico: fue una forma de representar cómo, en cada rincón de la universidad, nuestras emociones están presentes, acompañándonos silenciosamente en nuestro viaje académico y personal.

2. Objetivos

Objetivo General

Representar visualmente las emociones que experimentan los estudiantes universitarios durante su vida académica, mediante la creación de un cortometraje animado que integra personajes inspirados en la película *Intensamente*, insertados en escenarios reales de la universidad UTEC, utilizando técnicas avanzadas de computación gráfica como blending de imágenes, transferencia de color y animación 3D.

Objetivos Específicos

- **Representar las emociones universitarias mediante personajes animados:** Inspirarse en la estética y concepto emocional de la película *Intensamente* para diseñar personajes que encarnen emociones comunes en la vida universitaria como alegría, miedo, tristeza, ira y ansiedad. Estos personajes fueron utilizados para construir una narrativa que refleje visualmente el impacto emocional del entorno académico.
- **Implementar técnicas de blending y transferencia de color:** Aplicar algoritmos de Poisson Blending (tanto en su versión clásica como en su variante de gradientes mixtos), Alpha Blending y Color Transfer para lograr una integración visual natural entre los personajes y los escenarios reales. Estas técnicas permiten que los personajes animados se inserten armónicamente en fotografías reales, adaptándose tanto al color como a la textura del entorno.
- **Aplicar técnicas de movimiento en modelos 3D:** Utilizar herramientas de inteligencia artificial como Plask para capturar y transferir movimientos humanos reales a personajes tridimensionales. Esta técnica, conocida como motion tracking y motion transfer, permitió dotar derealismo a las animaciones, las cuales fueron integradas en una escena mediante Unity. Este proceso hizo posible la producción de secuencias animadas que reflejan interacciones naturales entre los personajes y el entorno universitario.

3. Marco Teórico

En este proyecto usamos tres técnicas principales para combinar o modificar imágenes: Poisson Image Blending, Color Transfer y Alpha Blending. Cada una tiene un propósito y

comportamiento diferente cuando se trata de mezclar o adaptar imágenes.

3.1. Poisson Image Blending

Es una técnica propuesta por Pérez, Gangnet y Blake [1] para realizar ediciones locales en imágenes de forma continua y natural, evitando bordes visibles. A diferencia del *copy-paste* tradicional, esta técnica no copia directamente los valores de color de una imagen fuente, sino sus **gradientes**, preservando así bordes, texturas y detalles estructurales al insertarlos en una imagen destino. Además, se tomaron referencias del material de clase dictado por Efros [2] y Gkioulekas [3] de Carnegie Mellon University. También, la documentación de proyectos ya existentes cubriendo este tema como los hechos por Aebi [4], Segal [5], Hahn [6] y Tao [7].

Formulación del problema

Dado un dominio Ω que representa la región seleccionada de la imagen fuente, queremos encontrar una función u (la imagen resultante en Ω) tal que su gradiente coincida lo más posible con el de la imagen fuente f , mientras que los valores en el borde $\partial\Omega$ coincidan con los de la imagen destino g :

$$\min_u \int \int_{\Omega} \|\nabla u - \nabla f\|^2 \quad \text{sujeto a } u|_{\partial\Omega} = g|_{\partial\Omega} \quad (1)$$

Este problema de minimización tiene una solución única que satisface la ecuación de Poisson:

$$\Delta u = \Delta f \quad \text{en } \Omega, \quad u = g \quad \text{en } \partial\Omega \quad (2)$$

donde Δ es el operador Laplaciano y ∇ representa el operador gradiente.

Discretización

En imágenes digitales, esta ecuación se resuelve sobre una malla de píxeles. Usando diferencias finitas, el Laplaciano de un píxel (i, j) se aproxima por:

$$\Delta u(i, j) \approx 4u(i, j) - u(i + 1, j) - u(i - 1, j) - u(i, j + 1) - u(i, j - 1) \quad (3)$$

Esto nos lleva a un sistema lineal para cada píxel $(i, j) \in \Omega$:

$$4u(i, j) - \sum_{(k,l) \in \mathcal{N}(i,j)} u(k, l) = \sum_{(k,l) \in \mathcal{N}(i,j)} [f(i, j) - f(k, l)] \quad (4)$$

donde $\mathcal{N}(i, j)$ es el conjunto de vecinos (arriba, abajo, izquierda, derecha) de (i, j) .

Forma matricial

El sistema anterior se puede expresar como una matriz dispersa:

$$A\mathbf{u} = \mathbf{b} \quad (5)$$

- A es una matriz esparsa que representa las conexiones entre píxeles vecinos.
- \mathbf{u} es el vector de incógnitas (valores de la imagen resultante en Ω).

- \mathbf{b} contiene las diferencias de gradiente y los valores de frontera.

Debido a la naturaleza dispersa y positiva definida de A , se utilizan solvers iterativos como *Jacobi*, *Gauss-Seidel* o *multigrid* para obtener una solución eficiente.

Interpretación

Desde el punto de vista energético, la solución u minimiza la diferencia entre su gradiente y el gradiente de la fuente f , bajo la condición de coincidir con la imagen destino en los bordes. Esto asegura que se preserven los detalles internos del objeto copiado, pero adaptando su color global e iluminación al nuevo contexto, logrando así buenos resultados.

Caso de gradientes mixtos

Para lograr una integración aún más realista, se puede usar una técnica tal que en cada píxel, se selecciona el gradiente de mayor magnitud entre el de la fuente y el del destino:

$$\vec{v}(x) = \begin{cases} \nabla g(x), & \text{si } |\nabla g(x)| > |\nabla f(x)| \\ \nabla f(x), & \text{en otro caso} \end{cases} \quad (6)$$

Esto se conoce como **Mixed Gradient Poisson Blending**, y permite preservar detalles importantes tanto de la fuente como del destino (por ejemplo, en objetos parcialmente transparentes o con textura del fondo).

Aplicación

Este método permite insertar objetos complejos con bordes irregulares en nuevos fondos sin necesidad de realizar selecciones muy precisas.

3.2. Color Transfer

La técnica de *Color Transfer*, propuesta por Reinhard et al. [8], busca cambiar los colores de una imagen fuente para que se vean como los de una imagen de referencia. Pero en lugar de copiar directamente los valores de color, lo que hace es copiar el estilo del color: es decir, cómo están distribuidos los colores (promedios y variaciones). Para que este proceso funcione bien, se transforma la imagen a un espacio de color especial llamado $\alpha\beta$, en el cual los canales están decorrelacionados y se pueden modificar por separado.

Motivación

Imagina que se tiene una imagen tomada con luz blanca, y se quiere que tenga el mismo tono cálido que otra imagen tomada al atardecer. O quizás se quiere que varias fotos parezcan parte de la misma escena, aunque fueron tomadas con diferentes cámaras o condiciones de luz. La idea detrás del *Color Transfer* es resolver este problema automáticamente, sin tener que ajustar colores manualmente. Para lograrlo, lo primero es convertir la imagen a un espacio de color que se parezca más a cómo percibe el ojo humano.

Transformaciones de color

El proceso comienza transformando la imagen de su espacio original (RGB) a un espacio donde el color y la luminancia están mejor separados. Este proceso incluye varios pasos:

1. **De RGB a LMS:** Primero, se transforma la imagen del espacio RGB al espacio LMS, que simula cómo los conos del ojo humano responden a la luz. La transformación es:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0,3811 & 0,5783 & 0,0402 \\ 0,1967 & 0,7244 & 0,0782 \\ 0,0241 & 0,1288 & 0,8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (7)$$

2. **De LMS a $l\alpha\beta$:** Luego se aplica logaritmo a cada canal (para simular cómo percibimos diferencias de intensidad) y se aplica una transformación lineal que nos lleva al espacio $l\alpha\beta$ [?]:

$$\begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3} \\ 1/\sqrt{6} & 1/\sqrt{6} & -2/\sqrt{6} \\ 1/\sqrt{2} & -1/\sqrt{2} & 0 \end{bmatrix} \begin{bmatrix} \log L \\ \log M \\ \log S \end{bmatrix} \quad (8)$$

En este nuevo espacio, l representa la luminancia (brillo), y α y β representan componentes cromáticos (verde-rojo y azul-amarillo). Lo importante es que estos canales están decorrelacionados, lo cual permite modificarlos de forma independiente.

Transferencia estadística

Una vez en el espacio $l\alpha\beta$, el truco es: igualar la media y la desviación estándar de los colores de la imagen fuente con los de la imagen de referencia. Para cada canal, se realiza lo siguiente:

1. Se calcula la media y desviación estándar de cada canal en la imagen fuente (S) y en la imagen objetivo (T):

$$\mu_S = \mathbb{E}[S], \quad \sigma_S = \text{std}(S) \quad (9)$$

2. Luego, se normaliza los valores de la imagen fuente:

$$S' = \frac{S - \mu_S}{\sigma_S} \quad (10)$$

3. Y luego se ajusta esa normalización con los valores de la imagen objetivo:

$$S_{\text{new}} = S' \cdot \sigma_T + \mu_T \quad (11)$$

Así, los colores finales de la imagen fuente tienen una distribución de color muy similar a los de la imagen de referencia, pero sin modificar la estructura de la imagen (bordes, objetos, etc.).

Reconversión a RGB

Después de aplicar el cambio de colores, se regresa al espacio RGB para visualizar la imagen. Para ello, se invierten las transformaciones anteriores:

- Se pasa de $l\alpha\beta$ a log-LMS (invirtiendo la matriz).
- Luego se aplican exponenciales para volver de log-LMS a LMS.
- Finalmente, se transforma LMS a RGB usando la siguiente matriz:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 4,4679 & -3,5873 & 0,1193 \\ -1,2186 & 2,3809 & -0,1624 \\ 0,0497 & -0,2439 & 1,2045 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix} \quad (12)$$

3.3. Alpha Blending

La técnica de **Alpha Blending** permite combinar una imagen con otra utilizando información de transparencia. Cada píxel de la imagen fuente tiene un valor de opacidad (canal alfa) que determina en qué medida debe mezclarse con el fondo. Es ampliamente utilizada en edición de imágenes, videojuegos, partículas, interfaces gráficas, etc. [9]

Formulación del problema

El principio básico de Alpha Blending es una interpolación lineal por píxel entre la imagen fuente F y la imagen de fondo B , ponderada por una máscara α :

$$I = \alpha \cdot F + (1 - \alpha) \cdot B \quad (13)$$

donde $\alpha \in [0, 1]$ es el valor normalizado del canal alfa (se divide entre 255 si está en formato entero). Este modelo se aplica por canal (R, G, B) de forma independiente.

Interpretación

- Si $\alpha = 1$, el píxel resultante es completamente de la imagen fuente (opaco).
- Si $\alpha = 0$, el píxel resultante es completamente del fondo (transparente).
- Si $0 < \alpha < 1$, se obtiene una mezcla proporcional entre ambos.

Esta interpolación permite crear bordes suaves, simular materiales translúcidos (como vidrio o humo) y superponer elementos visuales sin cortes bruscos.

4. Desarrollo

4.1. Guión

Desde un inicio se planteó que el corto sería una compilación de pequeños "skits" de 30 segundos de duración aproximadamente, cada uno centrado en una emoción en específico.

Con ello en mente, se realizó una lluvia de ideas y discusión de experiencias que hayamos tenido en la universidad, y se filtraron las emociones con las que se iba a trabajar, junto con sus respectivos escenarios.

Se hizo un primer boceto del guion a seguir para cada escena, y la forma en la que se iban a presentar los eventos cinematográficamente mediante un storyboard. Se optó por dividir la escena en dos partes: una mitad para la actuación en vivo de la situación presentada, y la otra mitad para la animación en 3D con los personajes de *Intensamente* reaccionando a la experiencia. Se decidió también hacer uso de imágenes con color transfer para complementar la animación e imágenes con image blending para hacer las portadas que determinan el inicio de cada escena.

Vista la cantidad de recursos necesarios para la elaboración del corto, se dividieron responsabilidades para conseguirlos todos.

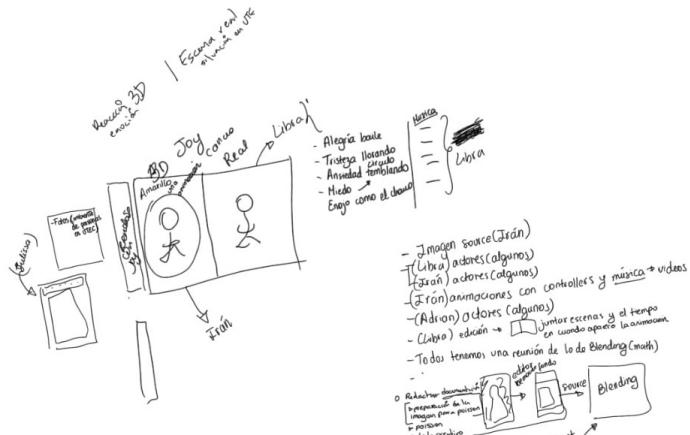


Imagen 1: Primer boceto de storyboard

4.2. Animación 3D

Se planteó realizar las técnicas de *motion capture* y *motion transfer* mediante una herramienta de IA denominada **Plask**. Esta herramienta nos facilitó realizar las animaciones mediante un flujo de trabajo.

Sketchfab: es una plataforma en línea que permite visualizar, compartir y descargar modelos 3D. Ofrece una amplia galería de assets creados por artistas y comunidades alrededor del mundo, muchos de ellos disponibles en formatos compatibles con herramientas de animación como GLB, FBX u OBJ. Para nuestro proyecto, se utilizó esta herramienta como fuente principal de modelos animados.

- **Obtención de assets 3D:** se descargaron assets (en formato **GLB**) que cuenten con huesos desde la página **SketchFab** para que el proceso de motion transfer sea lo más preciso posible.

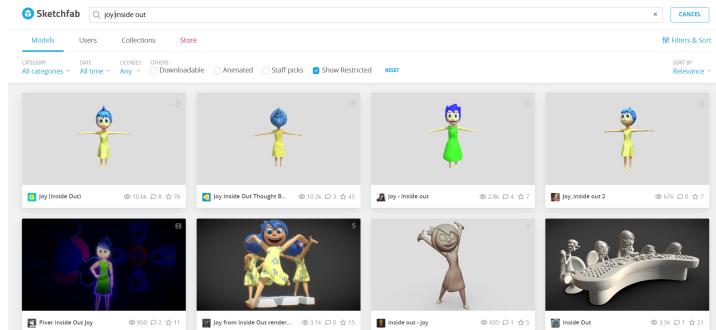


Imagen 2: Búsqueda de assets en Sketchfab

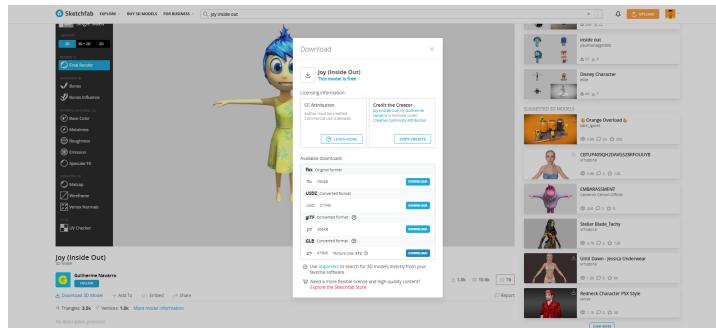


Imagen 3: Descarga de asset 3D usando GLB

Plask:

- **Creación de una escena:** se crea una escena en Plask de tal manera que podamos realizar la subida del asset 3D en formato GLB.

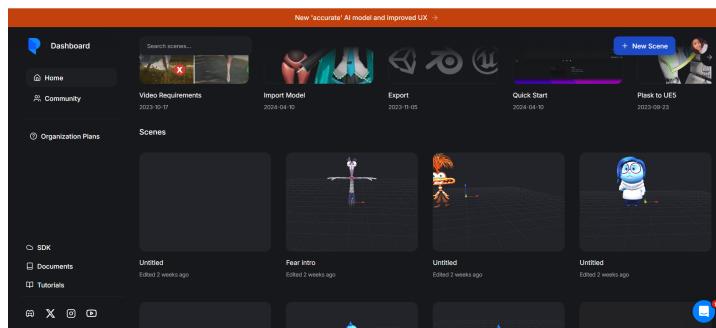


Imagen 4: Creando una escena nueva en Plask

- **Carga de asset en escena:** una vez en la escena, se sube el archivo GLB descargado de SketchFab en el entorno de trabajo.

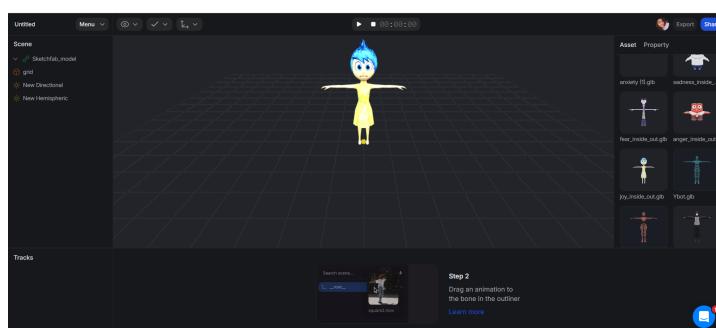


Imagen 5: Subida de asset 3D con huesos

- **Carga de video:** se capturó el movimiento humano a partir de un video grabado en formato MP4 de menos de 10 segundos mayormente.

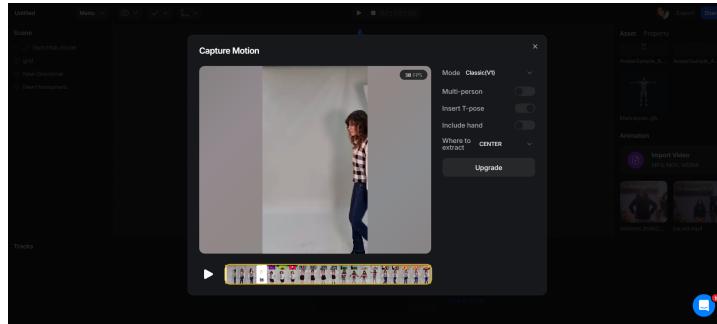


Imagen 6: Video simple de baile

- **Procesamiento con IA:** Plask aplicó su modelo de inteligencia artificial para detectar la postura y extraer automáticamente los puntos clave del esqueleto (pose estimation).
- **Generación del rig animado:** con los datos de pose, se generó una animación esquelética (rig) en formato 3D compatible con motores como Unity.

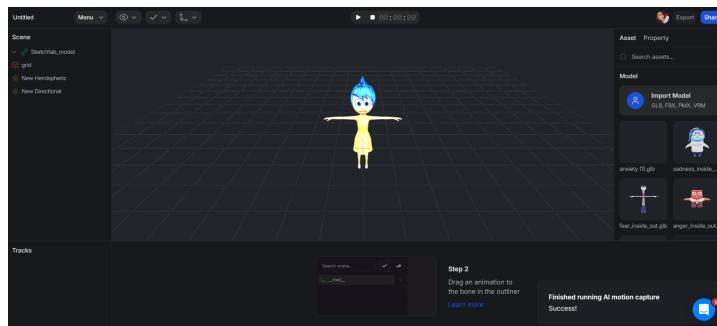


Imagen 7: Proceso de generación del rig animado

- **Motion transfer:** los movimientos extraídos se aplicaron a un personaje digital personalizado, permitiendo que el asset imitara el movimiento original.



Imagen 8: Motion transfer al asset de Joy

- **Exportación de la animación:** la animación final se exportó en formato FBX, ya que Unity trabaja mejor con este formato y realizó mejor el mapeo de textura por separado.

Unity: es un motor de desarrollo en tiempo real que permite integrar modelos 3D, aplicar animaciones y generar visualizaciones interactivas o videos exportables. En nuestro flujo de trabajo, Unity se utilizó para importar el personaje animado, configurar su escena y registrar la animación final desde la cámara.

A continuación, se muestra el proceso realizado:

- **Importación del personaje animado:** se importó el modelo 3D (.fbx) que contenía el rig generado previamente, y se incorporó a la escena en una posición central.



Imagen 9: Modelo 3D importado y posicionado en escena

- **Asignación de animador:** se vinculó un *Animator Controller* al modelo para manejar la animación transferida, permitiendo visualizarla dentro del entorno 3D de Unity.



Imagen 10: Configuración del Animator Controller del personaje

- **Asignación del clip animado:** mediante el editor de animación, se asignó el clip correspondiente a la animación generada por Plask, dentro del flujo base del controller.

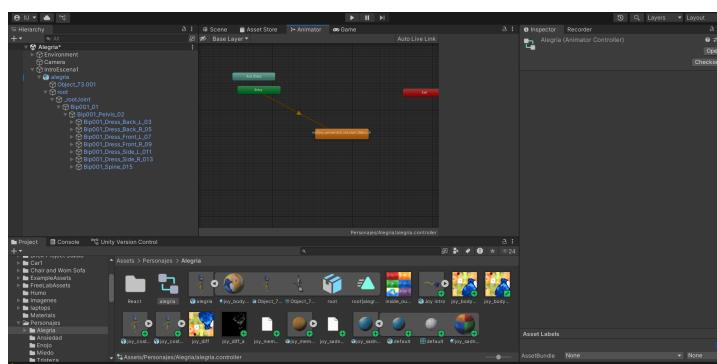


Imagen 11: Asignación del clip de animación al controlador

- **Grabación de la animación:** utilizando la herramienta de grabación de Unity (Unity Recorder), se exportó la animación en formato .mp4 a resolución Full HD desde la vista de cámara configurada.



Imagen 12: Configuración del Unity Recorder para exportar la animación

4.3. Implementación del blending

4.3.1. Selección de Imágenes

Para comenzar el proceso de blending, el usuario debe seleccionar dos imágenes: una fuente (de donde se copiará la región de interés) y una destino (donde se insertará dicha región). Esto se realiza mediante un cuadro de diálogo implementado con la librería `tkinter`, el cual permite escoger las imágenes desde el sistema de archivos local.

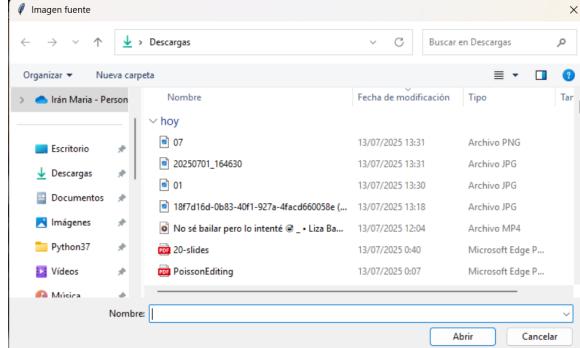


Figura 1: Imagen fuente: selección del archivo desde el sistema



Figura 2: Imagen fuente de Joy (427*831)

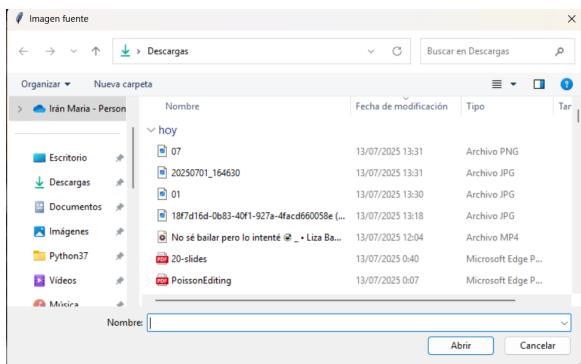


Figura 3: Imagen destino: selección del archivo desde el sistema



Figura 4: Imagen destino de UTEC (4000*3000)

4.3.2. Masking

La selección de la región a transferir se realiza de forma interactiva sobre la imagen fuente. El usuario dibuja manualmente un polígono cerrando el contorno de la región deseada. Este polígono se convierte internamente en una máscara binaria mediante la función `polygon()` de `skimage.draw`, que marca los píxeles pertenecientes a la región seleccionada. Esto permite mayor precisión y control al momento de definir el área a copiar.

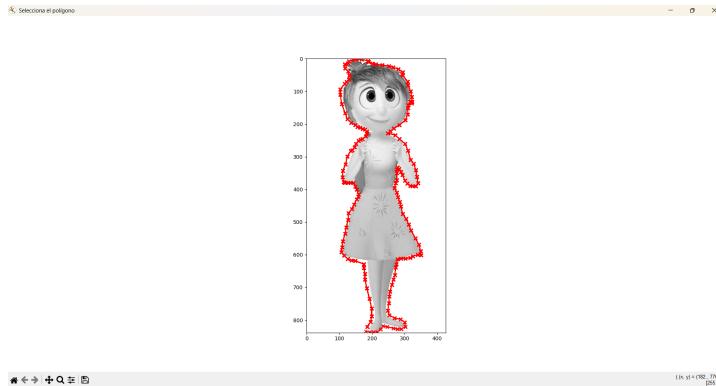


Imagen 13: selección de la región a transferir en la foto de Joy

4.3.3. Aplicación de las técnicas

Una vez definida la máscara y seleccionadas las imágenes, se procede a aplicar una de las tres técnicas disponibles:

- **Normal Poisson:** realiza el blending resolviendo el sistema de ecuaciones derivado de la ecuación de Poisson, replicando los gradientes de la imagen fuente en la región seleccionada.
- **Mixed Gradients:** a diferencia de la técnica anterior, esta variante mezcla los gradientes de la fuente y el destino, eligiendo en cada píxel el de mayor magnitud. Esto permite preservar detalles importantes tanto del objeto copiado como del fondo.
- **Alpha Blending:** se realiza una interpolación ponderada entre la imagen fuente y la de destino en la región de la máscara, usando un valor de transparencia (`alpha`) definido

por el usuario. Si $\text{alpha} = 1.0$, se copia directamente el píxel de la fuente; si $\text{alpha} = 0.0$, se conserva el fondo.

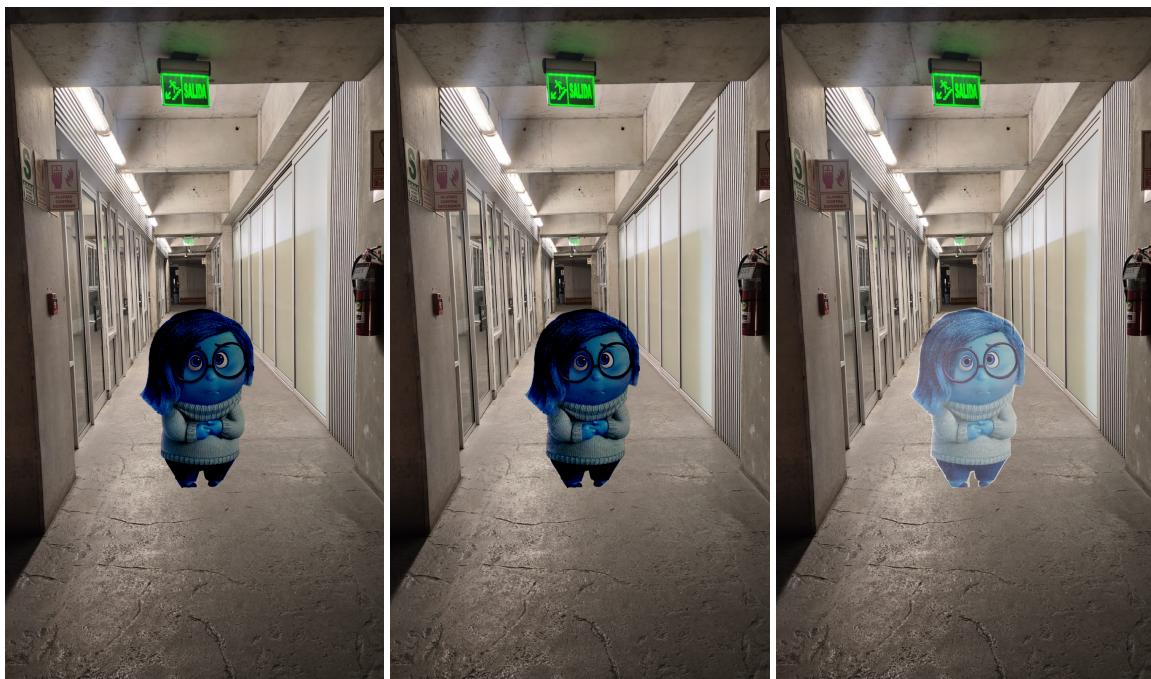


Figura 5: Blendings - Normal Poisson | Mixed Gradients | Alpha

Estas técnicas están encapsuladas en funciones modulares y reutilizables dentro del código fuente, lo cual permite mantener una estructura clara del flujo de trabajo. La solución se implementó canal por canal (R, G, B), y se reconstruye la imagen final combinando los resultados.

4.3.4. Environment

Para asegurar que el programa se pueda ejecutar en cualquier equipo sin necesidad de instalar manualmente todas las dependencias, se preparó un entorno virtual y un archivo `requirements.txt` con las librerías necesarias.

Este archivo contiene:

- **Pillow**: para carga y manipulación de imágenes en formato PNG/JPG.
- **NumPy**: para el manejo eficiente de matrices e imágenes como arreglos.
- **Matplotlib**: para visualización y captura de clics interactivos.
- **Tkinter**: para mostrar cuadros de diálogo de selección de archivos.
- **scikit-image (skimage)**: para generar máscaras desde polígonos.
- **scipy**: para crear matrices dispersas del sistema lineal.
- **pyamg**: solver multigrid eficiente para resolver ecuaciones de Poisson.

4.3.5. Implementación en código

La implementación de las técnicas de blending se desarrolló en Python utilizando librerías como NumPy, Pillow, scikit-image, matplotlib y pyamg. El enfoque se estructuró en módulos funcionales que permiten una interacción guiada con el usuario, facilitando el flujo completo desde la carga de imágenes hasta la generación de las imágenes resultado.

A continuación, se explican las funciones clave que componen la lógica del sistema:

- **Cargado y procesamiento de imágenes:** Se emplean funciones como las siguientes: `abrir_imagen_grises()` y `abrir_imagen_rgb()` para cargar imágenes en escala de grises y en formato RGB respectivamente. Estas son necesarias para calcular máscaras, extraer canales y preparar datos para el blending.
- **Selección de región y generación de máscara:** A través de la función `obtener_mascara()`, el usuario puede dibujar manualmente un polígono sobre la imagen fuente. Este polígono es convertido en una máscara binaria utilizando la función `polygon()` de la librería scikit-image, que determina qué píxeles deben copiarse a la imagen destino.
- **Blending tipo Poisson:** Para la técnica, se emplea la función `mezcla_poisson()`, que trabaja canal por canal (R, G, B). Esta llama internamente a:
 - `construir_sistema()`: construye la matriz dispersa y el vector b del sistema lineal a resolver. Este sistema se basa en la ecuación de Poisson y puede usar gradientes normales o mezclados (dependiendo del modo).
 - `resolver_sistema()`: resuelve el sistema lineal $A\mathbf{u} = \mathbf{b}$ usando un solver multigrid eficiente proporcionado por la librería pyamg.
 - `reconstruir_imagen()`: reconstruye la imagen destino insertando la región modificada con la solución del sistema.

Esta técnica logra una integración suave entre la región copiada y el fondo, conservando las texturas y bordes estructurales.

- **Mixed Gradient Poisson Blending:** En este modo, habilitado al seleccionar la opción 2, la función `construir_sistema()` compara los gradientes de la imagen fuente y destino en cada borde de píxel y elige el de mayor magnitud. Esto permite preservar tanto los detalles de la imagen copiada como la estructura del fondo, útil en escenarios complejos.
- **Alpha Blending:** La función `mezcla_alpha()` implementa blending clásico basado en un valor $\alpha \in [0, 1]$ ingresado por el usuario. La región definida por la máscara se combina como:

$$I = \alpha F + (1 - \alpha)B$$

donde F es la imagen fuente y B la imagen de fondo. Si $\alpha = 1$, se copia completamente la fuente; si $\alpha = 0$, se mantiene el fondo; y si $\alpha = 0,5$, se obtiene una mezcla uniforme.

- **Selección de ubicación:** La función `seleccionar_posicion()` permite al usuario hacer clic en la imagen destino para indicar el centro donde se colocará la región copiada. Esta coordenada se ajusta para centrar correctamente el recorte.
- **Recorte y alineamiento:** La función `recortar_region()` permite extraer la porción de la imagen destino que coincidirá en tamaño con la región copiada de la fuente. Esto garantiza que las operaciones se realicen sobre matrices del mismo tamaño.

- **Guardado y visualización del resultado:** Finalmente, la función `mostrar_y_guardar()` reconstruye la imagen RGB a partir de los tres canales modificados, guarda el resultado en un archivo .png y lo muestra al usuario.
- **Menú principal:** El bloque `main()` guía al usuario a través del flujo: selección del modo, carga de imágenes, generación de máscara, blending y guardado del resultado, lo cual hace más interactivo el programa.

4.4. Edición del Cortometraje

4.4.1. Recolección de Recursos Adicionales

Además de las animaciones 3D y las imágenes con blending, se requerían grabaciones de audio y de las escenas en persona para completar las escenas. Para grabarlas se emplearon celulares con cámaras y grabadoras de audio integradas. Se tuvo que recurrir a actores voluntarios en el caso de algunas escenas.

Para la banda sonora, se buscó en primera, música sin regalías. Estas fueron de selección propia y acorde a lo que la escena requería. Algo similar sucedió con los efectos especiales: se comenzó por buscar efectos gratuitos disponibles en línea. Para mantener la temática de Intensamente, también se hizo uso de música de la franquicia y se consiguieron algunas capturas de la película para emplearlas como parte del corto.

4.4.2. Edición del Vídeo

Con todos los recursos ya juntos, se procedió a editar el vídeo. Para ello se puede emplear cualquier programa de edición de vídeo (en nuestro caso empleamos VSCD Free Video Editor).

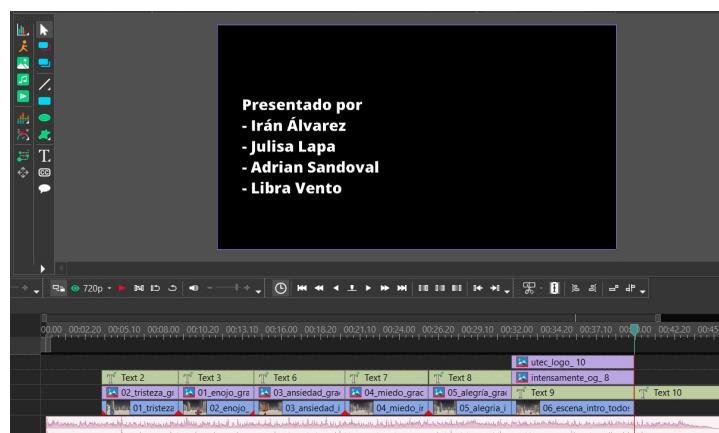


Imagen 14: Captura durante la edición del vídeo

Para evitar sobrecargar el aplicativo se dividió la elaboración del proyecto en tres partes principales:

- **Introducción:** presentación de los personajes.
- **Cuerpo:** escenas de cada escenario.
- **Fin:** montaje de imágenes de las emociones en distintos entornos de UTEC, acompañadas por voces de alumnos.

Para cada sección se hizo uso de los recursos que necesitaban y luego de una revisada grupal, se juntaron todas las partes en un solo video. Se añadieron algunos retoques finales, se exportó y se subió a YouTube.

5. Conclusión

5.1. Resultados y Reflexión

Este proyecto representó una oportunidad para combinar creatividad, narrativa y herramientas avanzadas de computación gráfica con el fin de explorar visualmente el universo emocional de los estudiantes universitarios. Inspirándonos en la película *Intensamente*, logramos insertar personajes animados en escenarios reales de la universidad, utilizando un conjunto de técnicas de *image blending*.

Durante el desarrollo, implementamos soluciones como *Poisson Blending*, *Alpha Blending* y *Color Transfer* para lograr integraciones visuales lo más naturales posibles. Asimismo, aplicamos técnicas de *motion transfer* con inteligencia artificial mediante Plask, y preparamos entornos de animación con Unity para producir el cortometraje final.

Más allá del resultado visual, el proyecto nos permitió comprender de manera práctica la importancia de los gradientes, las transformaciones de color, las máscaras de segmentación, y la lógica detrás del tratamiento de imágenes y modelos 3D. Cada componente técnico fue un desafío que nos llevó a aprender no solo herramientas, sino también a tomar decisiones de diseño digital.

Sin embargo, el proyecto también evidenció diversas limitaciones, desde las restricciones de las plataformas utilizadas hasta dificultades con el control del blending en entornos complejos. Estas observaciones abren oportunidades claras para futuras mejoras, como la implementación de segmentación automática, refinamiento en el pipeline de animación o la explotación de nuevas herramientas de IA para edición de video.

Finalmente, más allá del valor técnico, este trabajo nos permitió reflexionar sobre cómo las emociones, invisibles, pero constantes, pueden representarse visualmente y cómo la tecnología puede humanizarse a través de la narrativa. Logramos no solo hacer visible lo invisible, sino también construir una historia donde la emoción y la técnica se encontraron en un mismo espacio.

5.2. Limitaciones y Futuras Mejoras

Mientras desarrollamos el proyecto, encontramos ciertas problemáticas, específicamente con el tratamiento de las imágenes, las cuales detallaremos a continuación:

- **Calidad de las imágenes fuente:** En varios casos, las imágenes utilizadas como fuente no contaban con la resolución o calidad suficiente para lograr una integración visual natural. Esto generó pérdida de detalle durante el blending, especialmente en bordes. En el futuro, se debería trabajar con imágenes de alta resolución y con un tamaño de alto y ancho que no sobrepase al tamaño de la imagen target desde el inicio del flujo de trabajo.
- **Selección de imágenes de entrada:** Si bien Poisson Blending está diseñado para manejar casos en los que hayan mucha diferencia entre una imagen y otra [6], los mejores resultados se obtuvieron cuando la diferencia de colores en las imágenes fuente y destino eran menores. Es por este motivo que se añadió el uso de color transfer en la imagen fuente

previo al blending. Sin embargo, sigue presentando limitaciones, especialmente cuando la iluminación es muy fuerte, muy pobre o hay reflejos en la imagen. Queda como mejora buscar una manera más efectiva de atenuar los problemas en estos casos. Cabe mencionar que al momento de buscar soluciones para ello, ha de considerar el uso práctico se le está dando al blending en este caso y los recursos que se tienen disponibles. Poisson es efectivo y no requiere tanto poder computacional como lo haría un *Gaussian pyramid* o *Laplacian pyramid*, que funcionan mejor con una GPU a la mano [10].

- **Detección de bordes imperfecta:** Aunque el usuario podía trazar manualmente la región de interés, el proceso dependía en gran medida de su precisión al dibujar. Esto generaba contornos irregulares o bordes duros en la inserción. Una mejora viable sería implementar un algoritmo de segmentación automática.
- **Limitación en tiempo y duración del video en Plask:** La plataforma Plask, usada para el motion capture, tiene un límite en la duración del video (15 segundos) en su versión free, por lo que tuvimos que optar por la versión pro que tiene un límite mayor de tiempo. Sin embargo, ello aun así restringió la complejidad de las escenas animadas. Como mejora, se podría explorar alternativas como Rokoko Studio, que permiten mayor control y personalización.
- **Limitaciones del blending clásico en escenas complejas:** Las técnicas como Poisson Blending funcionan bien en regiones pequeñas con contraste moderado, pero generan puntos negros visibles cuando se insertan objetos grandes o en fondos con texturas complejas. Como mejora futura, se puede considerar el uso de técnicas más avanzadas como Seamless Cloning multiescala o blending guiado por redes neuronales.
- **Falta de control en el resultado final:** Si bien el sistema funcionaba de forma automática, no ofrecía al usuario una vista previa ni control sobre la intensidad del blending o el contraste del resultado. Una posible mejora sería implementar una interfaz gráfica que permita ajustar parámetros como opacidad, brillo o recorte final antes de exportar la imagen.
- **Imagen source fuera del bounding box de la imagen target:** Realizando las experimentaciones con las imágenes source y target, notamos que si se coloca la máscara (obtenida luego de trazar puntos en la imagen fuente) muy cerca al borde de la imagen destino, ocurría un error. El mensaje de error indicaba que los límites de la imagen fuente estaban fuera del alcance de la imagen destino, lo que causaba que el usuario corra el programa desde el inicio. Para mejorar ello, usaríamos la lógica de un bucle **while** que permita al usuario seleccionar otra parte de la imagen de destino para que no pierda el trabajo realizado con la máscara.

Referencias

- [1] A. B. Patrick Pérez, Michel Gangnet, “Poisson image editing,” 2003. [Online]. Available: <https://www.cs.jhu.edu/~misha/Fall07/Papers/Perez03.pdf>
- [2] A. Efros, “Image blending,” 2010. [Online]. Available: https://graphics.cs.cmu.edu/courses/15-463/2010_spring/Lectures/blending.pdf
- [3] I. Gkioulekas, “Image blending,” 2017. [Online]. Available: https://graphics.cs.cmu.edu/courses/15-463/2017_fall/lectures/lecture7.pdf
- [4] B. Aebi, “Cs129 project 2: Gradient domain fusion using poisson blending,” 2012. [Online]. Available: <https://cs.brown.edu/courses/cs129/results/proj2/baebi/>

- [5] H. Segal, "Poisson blending," 2022. [Online]. Available: <https://github.com/Haim-Segal/Poisson-Blending>
- [6] D. Hahn, "Image blending," 2017. [Online]. Available: https://ddavidhahn.github.io/194-26_Project3/
- [7] X. Tao, "Gradient domain fusion using poisson blending," N.A. [Online]. Available: <https://cs.brown.edu/courses/cs129/results/proj2/taox/>
- [8] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Computer Graphics and Applications*, vol. 21, no. 5, pp. 34–41, 2001. [Online]. Available: <https://ieeexplore.ieee.org/document/946629>
- [9] T. McREYNOLDS and D. BLYTHE, "Chapter 11 - compositing, blending, and transparency," in *Advanced Graphics Programming Using OpenGL*, ser. The Morgan Kaufmann Series in Computer Graphics, T. McREYNOLDS and D. BLYTHE, Eds. San Francisco: Morgan Kaufmann, 2005, pp. 185–209. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9781558606593500135>
- [10] J. S. Kim, M.-K. Lee, and K.-S. Chung, "Image blending techniques based on gpu acceleration," in *Proceedings of the 2018 International Conference on Image and Graphics Processing*, ser. ICIGP '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 106–109. [Online]. Available: <https://doi.org/10.1145/3191442.3191471>

Anexos

- **Carpeta drive del proyecto:**
<https://drive.google.com/drive/folders/1oGgVsM-y5tBTJR6KZps0uLNCjfKnjsU4>
- **Presentación:**
<https://www.canva.com/design/DAGrIfVjwss/ECgLTPeTjhuky80ysHSkEA>
- **Repositorio del proyecto:**
<https://github.com/iranalvarez27/IntensamenteEnUTEC.git>
- **Video del cortometraje:**
<https://youtu.be/zo7Fq73QGXs>