

Programación Avanzada

IIC2233 2023-2

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Joaquín Tagle - Francisca Cattán



Experiencia 3

Interfaces Gráficas II



Experiencia 3: ¿Qué vamos a hacer?

1. Volveremos a aplicar el patrón de diseño: *front-end/back-end*.
2. Crearemos una aplicación de múltiples componentes que interactúan entre ellos mediante el **uso de señales**.
3. Aplicaremos OOP y *overriding* para **personalizar QLabel**.
4. Incluiremos **conurrencia** a la aplicación para observar múltiples cambios simultáneos en el front-end.
5. Incluiremos **música** a la aplicación.

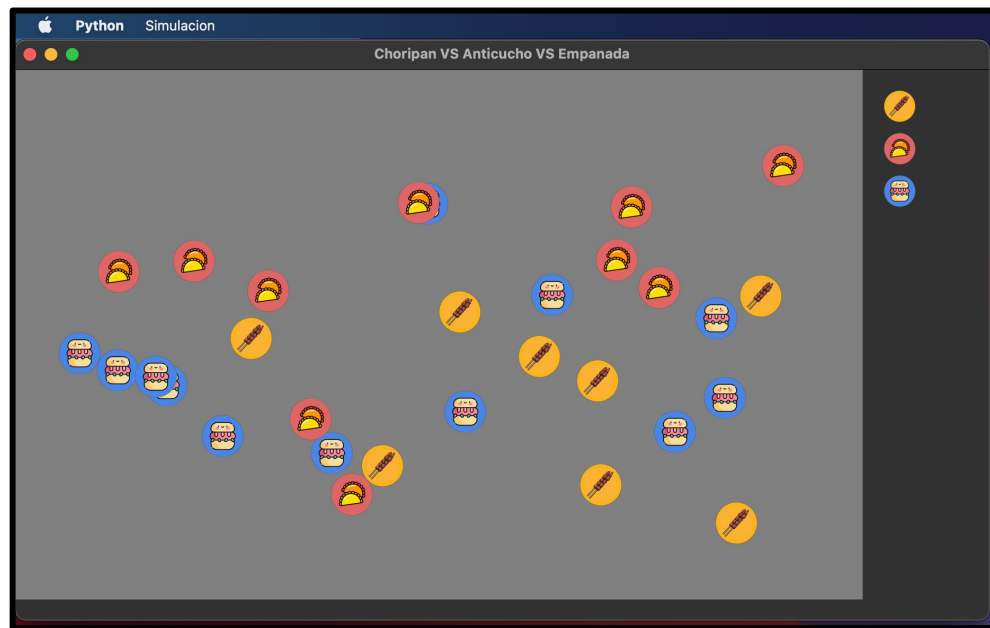
DCComida

Luego de unos terremotos, surge la interrogante, ¿qué es mejor? ¿la empanada, choripan o anticucho?

Para resolver esta interrogante, crearemos una simulación donde compiten estas 3 comidas.

¿Cómo lo lograremos?

Programaremos una simulación compuesta por tres componentes:



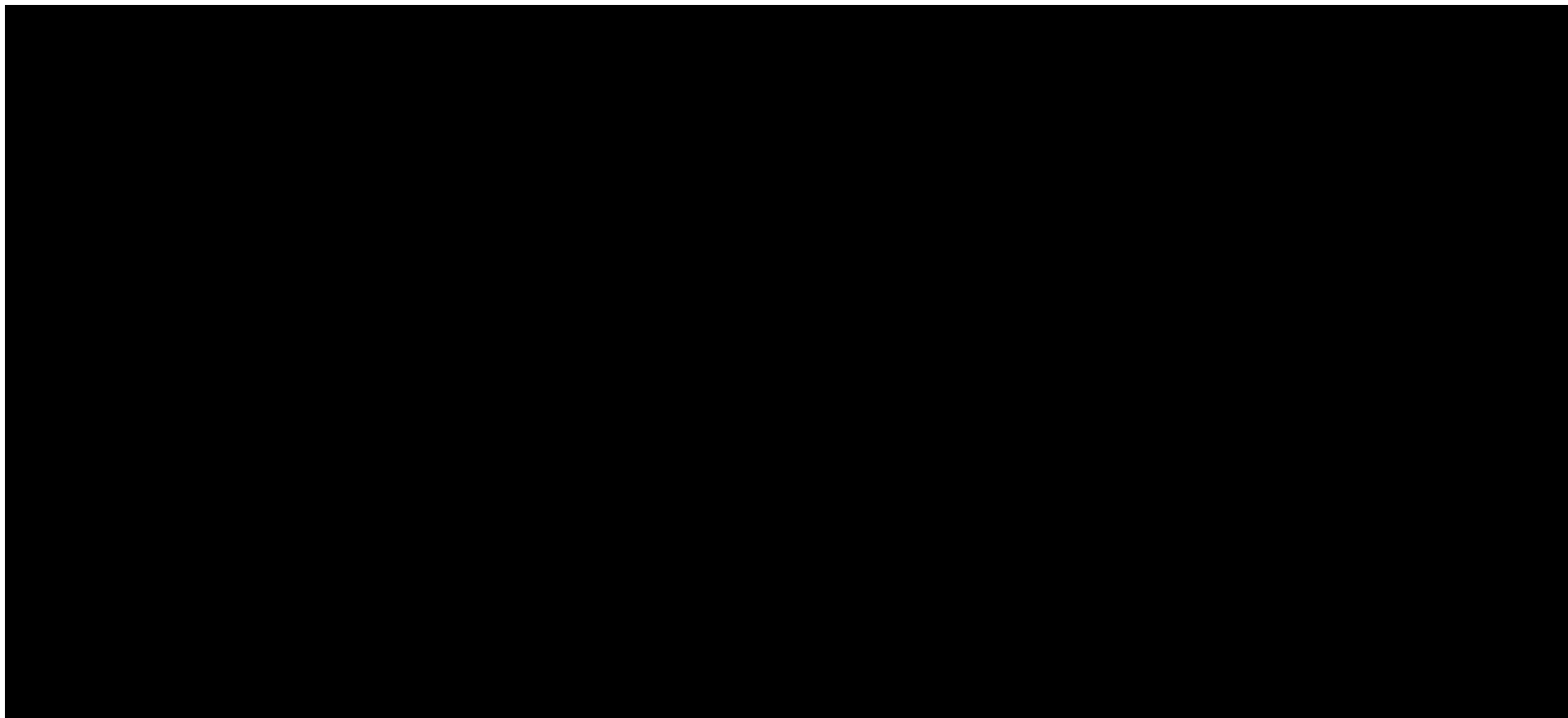
Frontend

Controlador encargado de
la simulación

Controlador de un Ícono

¿Cómo lo lograremos?

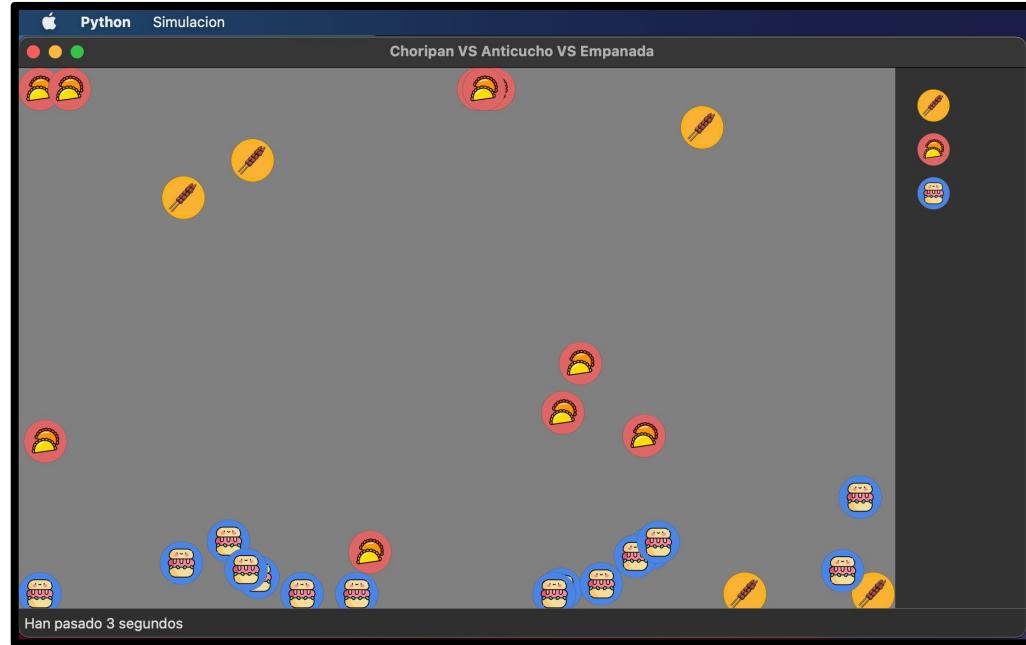
Veamos una mini demo



Front-end

Está compuesto por:

- Muchos QLabels.
- *Background* clickeable.
- QLabels clickeables.
- Barra de menú para empezar la simulación.
- Barra de estado para indicar el tiempo.
- Música de fondo.



Controlador lógico

- Se encargan de procesar toda la información lógica del programa.
 - Crear todos los íconos
 - Comenzar la simulación
 - Actualizar el tiempo que llevamos

**Controlador
encargado
de la lógica**

Controlador de un Ícono

- Se encargan de procesar toda la información lógica de 1 ícono.
 - Almacena y actualiza su poción
 - Busca a un ícono enemigo más cercano para convertirlo
 - Avisa al controlador lógico de cada colisión
 - Avisa al *frontend* cada vez que mueve el ícono



Controlador
de un Ícono

¿Cómo lo lograremos?

Estos componente se encuentran incompletos, por lo que deberemos **completar los métodos** faltantes de cada componente:

Parte 1 - Front-end

1. Personalizar 2 *QLabel*
2. Definir nuestra barra de menú
3. Crear y reproducir música

Parte 2 - Controlador Icono

1. Crear *QTimer* para permitir el movimiento del ícono

Parte 3 - Controlador lógico

1. Crear *QTimer* para actualizar tiempo y emitir señal correspondiente.
2. Crear Icono cada vez que se haga click en la pantalla.
3. Completar método “empezar” para dar inicio a la simulación.
4. Completar método “choque” para cuando un ícono colisiona con otro.

¿Cómo lo lograremos?

Estos componente se encuentran incompletos, por lo que deberemos **completar los métodos** faltantes de cada componente.

Finalmente, rellenar el archivo **main.py** para que **conecte** las distintas señales (si, son tan importantes que lo volveremos a hacer en esta experiencia)

¿Qué tenemos? (*Front-end*)

IconoClickeable

```
↺ senal_click: str

! mousePressEvent(
    event: QMouseEvent
)
```

FondoClickeable

```
↺ senal_click: int int

A tipo: str

! mousePressEvent(
    event: QMouseEvent
)
```

VentanaSimulacion


```
↺ senal_click_pantalla: int int str
↺ senal_empezar: null














A ancho_simulacion: int
A largo_simulacion: int
A tamaño_icono: int
A ultimo_seleccionado: str or null
A background: FondoClickeable
A labels: dict[str] = QLabel
A pixmaps: dict[str] = QPixmap
A self.media_player: QMediaPlayer


! empezar_simulacion()
✓ agregar_icono(x: int, y: int)
✓ seleccionar(tipo: str)
✓ actualizar_tiempo(segundos: int)
✓ aparecer_icono(id_icono: int, x: int, y: int, tipo: str)
✓ actualizar_icono(id_icono: int, nuevo_tipo: str)
✓ mover_icono(id_icono: int, x: int, y: int)
```

¿Qué tenemos? (*Back-end*)





Icono







 `senal_choque: int str`





 `id: int`
 `ancho_simulacion: int`
 `largo_simulacion: int`
 `tamaño_icono: int`
 `tipo: str`
 `senal_mover: pyqtSignal`
 `tamaño_icono: int`
 `ultimo_seleccionado: str or null`
 `x: int (property)`
 `y: int (property)`
 `centro_x: int (property)`
 `centro_y: int (property)`
 `otros_enemigos: list[Icono]`


 `mover()`

ControladorLogico

 `senal_mover_icono: int int int`
 `senal_actualizar_icono: int str`
 `senal_aparecer_icono: int int int str`
 `senal_actualizar_tiempo: int`

 `ancho_simulacion: int`
 `largo_simulacion: int`
 `tamaño_icono: int`
 `segundos: int`
 `empece: bool`
 `iconos: dict[int] = Icono`

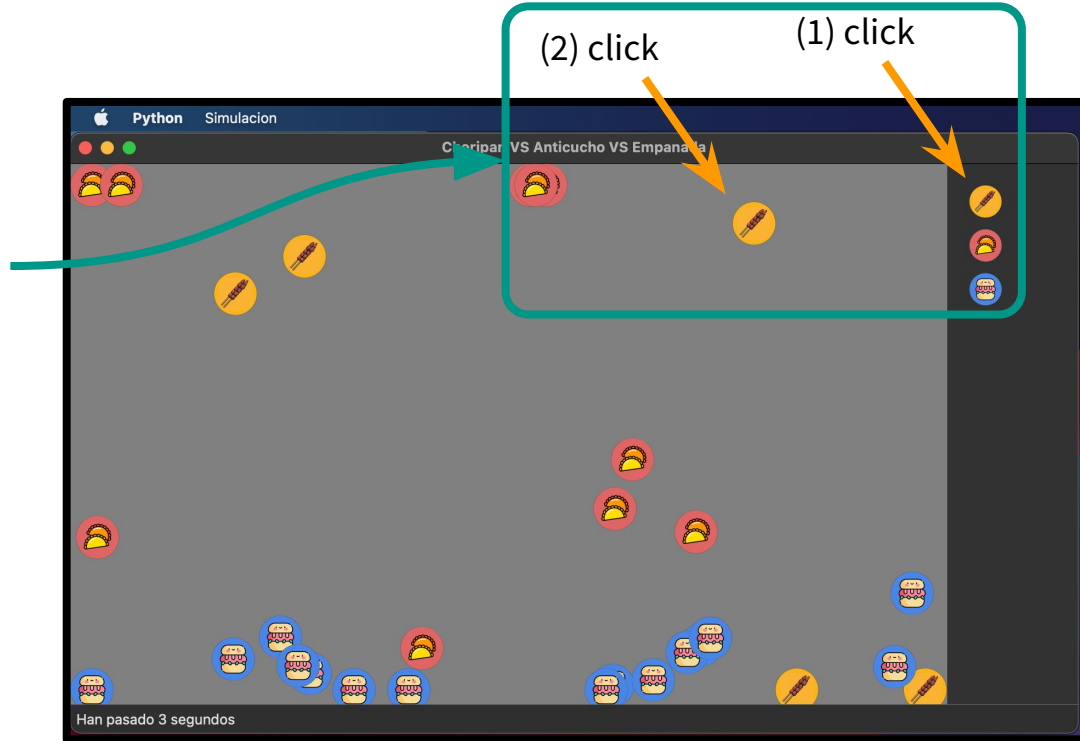
 `actualizar_tiempo()`
 `empezar()`
 `click_pantalla(x: int, y: int, tipo: str)`
 `choque(enemigo_id: int, tipo: str)`

 `poblar_random()`

Parte 1: *Front-end*

Completar **ventana.py** para:

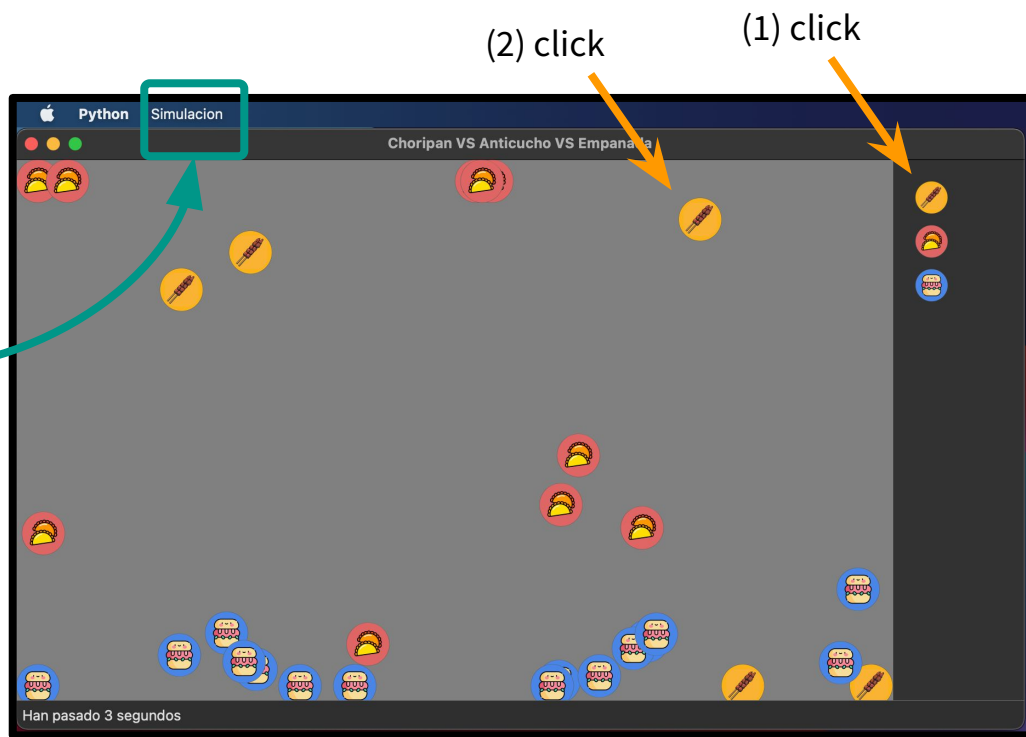
1. **Personalizar 2 QLabel para poder hacer click y poner más iconos**
2. Definir nuestra barra de menú
3. Crear reproductor y reproducir música cuando comenzamos la simulación



Parte 1: *Front-end*

Completar **ventana.py** para:

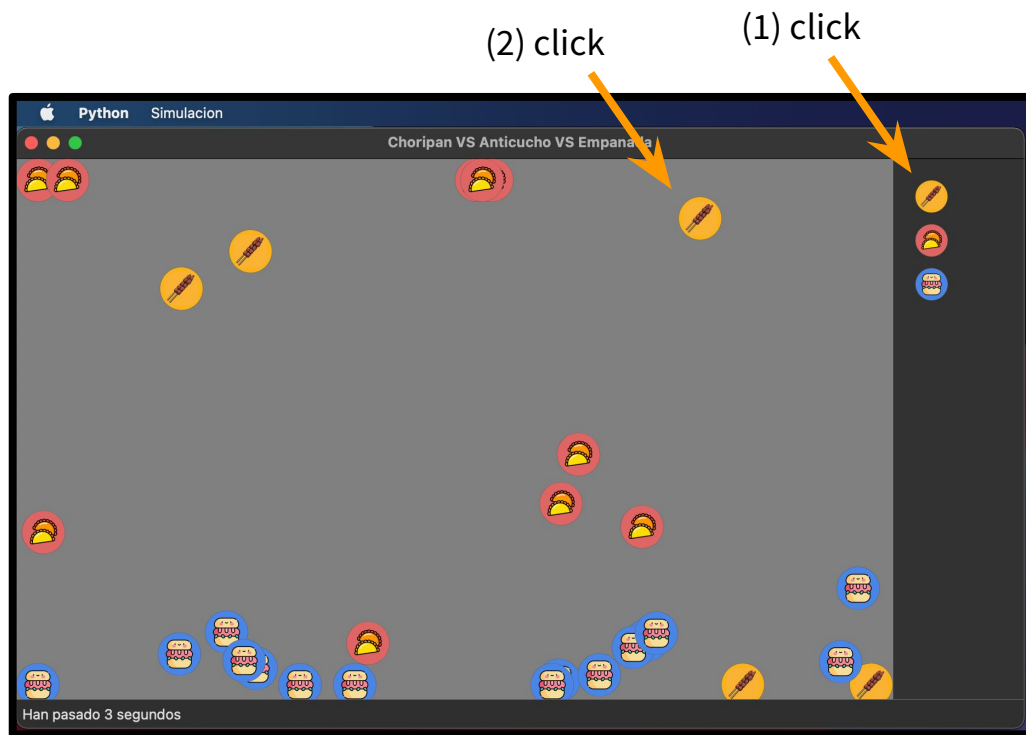
1. Personalizar 2 QLabel para poder hacer click y poner más iconos
2. **Definir nuestra barra de menú**
3. Crear reproductor y reproducir música cuando comenzamos la simulación



Parte 1: *Front-end*

Completar **ventana.py** para:

1. Personalizar 2 QLabel para poder hacer click y poner más iconos
2. Definir nuestra barra de menú
3. **Crear reproductor y reproducir música cuando comenzamos la simulación** 🇨🇱 🔊



Parte 1: *Front-end*

Vamos al código



Parte 2 y 3: *Back-end*

Parte 2 - *Controlador Icono*

1. Crear *QTimer* para permitir el movimiento del ícono

Parte 3 - *Controlador lógico*

1. Crear *QTimer* para actualizar tiempo y emitir señal correspondiente.
2. Crear Icono cada vez que se haga click en la pantalla.
3. Completar método “empezar” para dar inicio a la simulación.
4. Completar método “choque” para cuando un ícono colisiona con otro.

Parte 2 y 3: *Back-end*

Vamos al código



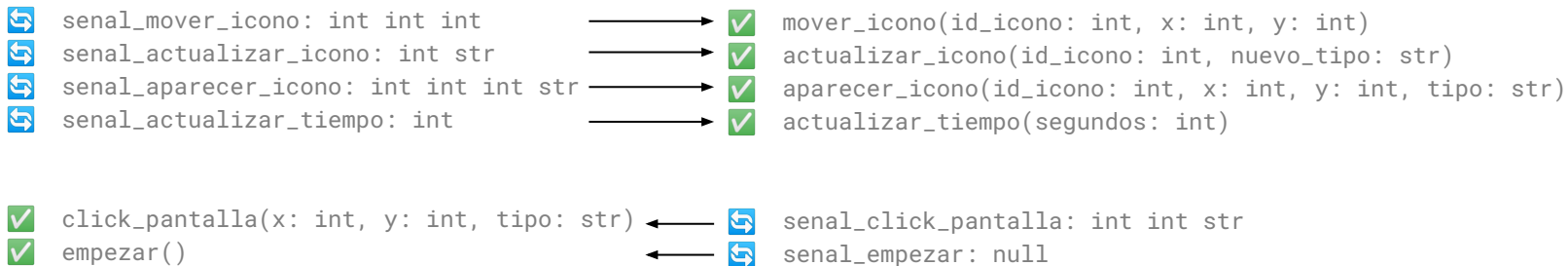
Parte 4: Señales

Los componentes visuales y lógicos ya se encuentran completos, solo falta:

Conectar las señales en el **main** entre *back-end* y *fron-tend*

ControladorLogico

VentanaSimulacion



Parte 4: Señales


Vamos al código



¿Eso es todo amigos?

Se puede mucho más, ideas para practicar

- Agregar una acción en la barra de menú para reiniciar la simulación.
- Permitir pausar la música.
- Detener el tiempo cuando ya ganó una comida
- Permitir eliminar íconos ya ingresados.
- Permitir agregar íconos en la mitad de la simulación y que se unan a la batalla.

Si todavía queda tiempo, intenten hacer uno de estos con la guía de su profesor, sino inténtelo en casa y/o con algún amigo/a .

Programación Avanzada

IIC2233 2023-2

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Joaquín Tagle - Francisca Cattán

