

Program Flow Description:

Image Initialization and Loading:

- The process begins by uploading a musical score image provided by the user.
- The image is converted to grayscale and binarized to facilitate element detection.

Element Detection Using Templates:

- The `locate_images` function is used to identify and locate different musical elements (staves, sharps, flats, notes) within the image.
- This function is based on the correspondence of templates with the image to be analyzed, adjusting the scale and applying a confidence threshold to identify matches.

Processing of Detections:

- The positions of the detections are stored in lists of rectangles (`Rectangle`), which represent the areas of the image where matches with the templates have been found.
- Detections are filtered and grouped to remove duplicates or irrelevant matches, using techniques such as `merge_recs`.

Grouping and Classification of Notes:

- The detected notes are grouped according to their position on the staff, taking into account modifying elements such as sharps and flats.
- The `Note` class is responsible for assigning the pitch and name of the musical note, adjusting the pitch if sharps or flats are detected on the same staff.

MIDI File Generation:

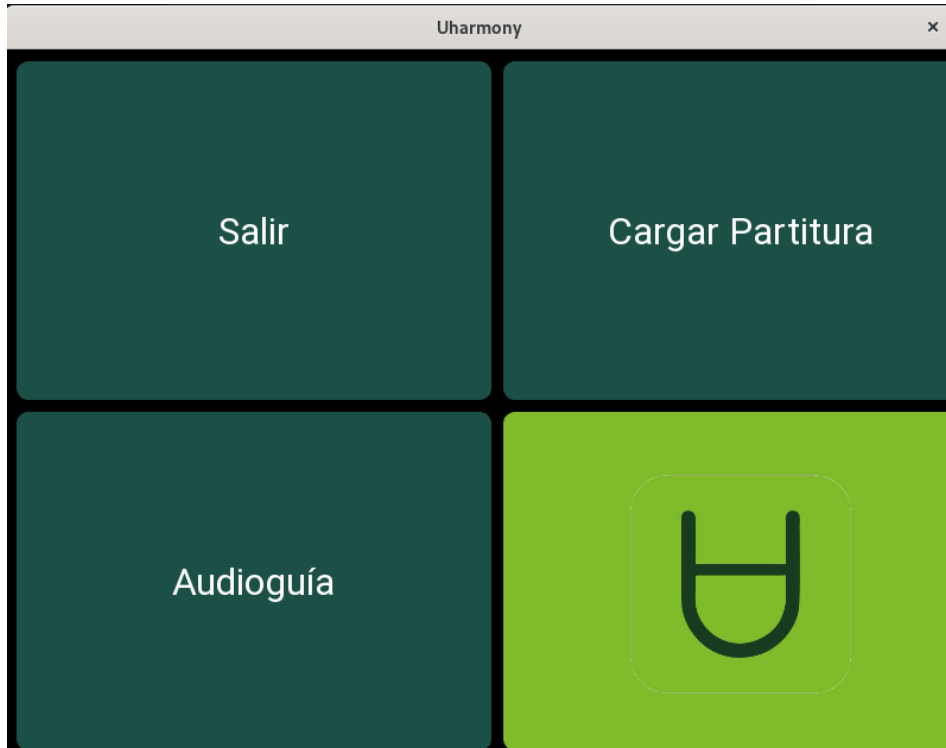
- Based on the grouped and classified notes, a MIDI file representing the detected musical score is generated.
- The MIDI file is converted to an audio format (MP3) for playback.

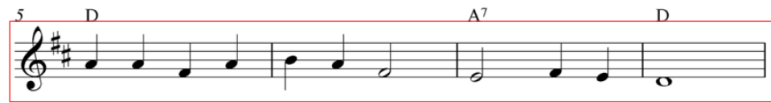
Used Modules:











- `random.randint()`: This function from the `random` library generates random integers within a specified range. In this case, it could be used to generate random positions or random values that are needed during image processing.









- `midutil.MIDIFile3`: This library allows you to generate MIDI files from musical data. In the code, it is used to create the final MIDI file that will represent the detected partition.
- `pydub`: The `pydub` library is used to manipulate and play audio files, such as WAV or MP3. This can be useful for generating or playing sounds corresponding to notes detected in the partition.
- `plyer.filechooser`: The `plyer` library provides an interface to access native device functionality, such as the file chooser. In this case, it is used to allow the user to select the partition image to process.
- `sys`: The `sys` library provides access to some variables used by the Python interpreter and to functions that interact with the operating system. It can be useful for handling command line arguments or performing system operations.
- `math`: The `math` library provides access to standard mathematical functions, such as trigonometric, exponential, and logarithmic. It can be useful for performing geometric and mathematical calculations necessary during image processing.
- `numpy`: `numpy` is a Python library specialized in handling multidimensional arrays and arrays. It is widely used in image processing and numerical calculations.
- `cv2` (OpenCV): OpenCV is a computer vision and image processing library. It is used extensively in this code to detect and process the partition elements in the image.
- `os`: The library provides you with a way to interact with the operating system, such as accessing files and directories, or executing system commands.
- `subprocess`: This library allows you to run external processes from Python, which can be useful for integrating the code with other tools or applications.
- `Pygame`: `Pygame` is a Python library that makes it easy to develop games and multimedia applications. In this case, it could be used to play the sounds corresponding to the notes detected in the partition.

This is the interface, super simple. 3 buttons. In loading score, the image is selected and to carry out all the processing, then it automatically plays and takes you to the second interface window (to play again or go to the beginning). The other button that says Audioguide is to play a preloaded mp3 audio.





Símbolo	Etiqueta	Plantilla
Negra	quarter	
	solid-note	
Blanca	half-line	
	half-note- line	
	half-note-space	
	half-space	
Redonda	whole-line	
	whole-note-line	
	whole-note-space	
	whole-space	

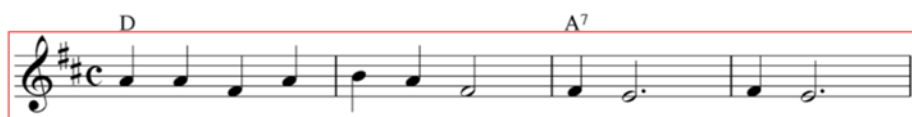
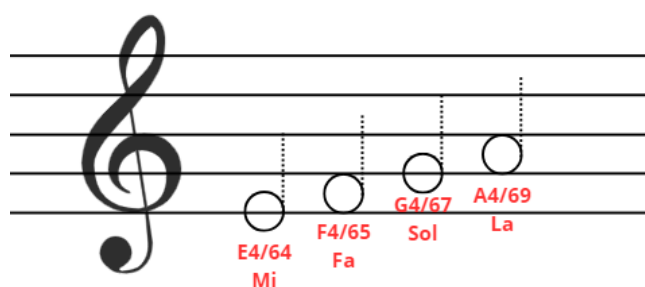
Sostenido	f-sharp	
	sharp	
Bemol	flat-line	
	flat-space	
Líneas	staff	
	staff2	
	staff3	
Silencio de blanca	bar-rest	

In order for the application to play musical scores correctly, the user is required to enter treble clef scores. The range of notes used is shown in Table X. This table provides information about the notes, their respective octaves, and their MIDI notation.

NOTA	OCTAVA	NOTACIÓN EN MIDI
FA	F6	89
MI	E6	88
RE	D6	86
DO	C6	84
SI	B5	83
LA	A5	81

SOL	G5	79
FA	F5	77
MI	E5	76
RE	D5	74
DO	C5	72
SI	B4	71
LA	A4	69
SOL	G4	67
FA	F4	65
MI	E4	64
RE	D4	62
DO	C4	60
SI	B3	59
LA	A3	57

This image shows the location of the notes corresponding to the treble clef on the staff, as well as their MIDI notation.



Easy flute

Camptown Races

Trad.

The image shows a musical score for the song "Camptown Races" on an easy flute. The score is written on three staves, each with a treble clef and a key signature of one sharp (F#). The time signature is common time (C). The first staff contains measures 1 through 4, with a D major chord above measure 1 and an A7 chord above measure 3. The second staff contains measures 5 through 8, with D major chords above measures 5 and 8, and an A7 chord above measure 6. The third staff contains measures 9 through 12, with D major chords above measures 9 and 12, and a G major chord above measure 10. Red square markers are placed below the notes in measures 2, 3, 6, and 7. The score is presented in a clean, simple format with a light blue border.