Fantoccini Specification v1.0.0

# Text Description automatically generated with medium confidence

# Vision

“Fantoccini is an animation studio designed to allow creators the freedom to create natural, fluid, and highly detailed animations in a fraction of the time required by traditional keyframe-based systems.”

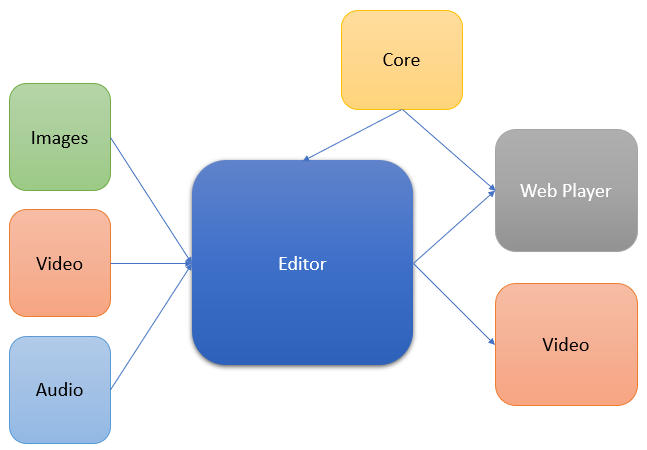
# Animation Through Realtime Kinematic Input

Fantoccini provides a rich set of mapping between input sources such as mouse, keyboard, and controllers to allow creators the ability to progressively layer real-time input to control any animatable property. Fantoccini treats animation more like a live recording than a static manipulation of data. This enables superior control and creative expression with reduced friction.

# High Level Breakdown

Fantoccini is broken down into several high-level components. The Core functionality provides playback of scene and animation data, whereas the Editor provides capability for recording and editing in real-time.

Media such as images, video and audio can be imported and manipulated in the Editor to create animation resources and elements.



Both the Editor and the Web Player share the Core functionality for real-time playback. Exporting video allows creators the ability to publish their work on social media platforms such as YouTube, whereas the Web Player allows creators the ability to embed their animations into websites and control the animations programmatically.

# Internal Core Concepts & Elements

Fantoccini is made up of the following high-level concepts:

## Project

A Project is the top-level object used to create an animation.

* It has a title
* It has a global fps
* It holds a single Timeline
* It holds multiple Scenes
* It has a single visual output which comes from one Scene at any given time, this can be animated to cut between Scenes at any time

## Ticker

A Ticker drives the animation by controlling a frame rate and updating the Project when new frames occur.

* It uses extra precision to ensure an accurate framerate

## Timeline

A Timeline provides a global time range for the Project. It holds multiple Scenes which are all driven from the one timeline and ticker.

* It holds multiple Scenes
* It holds multiple audio clips
* It drives all objects from the Ticker

## Scene

A Scene holds the actual animatable objects. Each Scene can have its own composition of objects while the main Timeline is progressing. Scene’s can contain multiple Cameras to capture different visuals and the Scene will be outputting one of those Cameras as its primary output. This choice of Cameras can be animated like any other Property.

* It holds Sprites
* It holds multiple Cameras
* It uses a single Camera for current Scene visual output to Project
* It holds Sprites

## Camera

A Camera is an object which captures a visual area of a Scene.

* It has a position and view size
* It has a zoom value

## AnimatableProperty

An AnimatableProperty is a property of either a numeric, string or boolean type which has a DataChannel to hold an array of values. These properties can have a default static value, otherwise whenever a value is accessed a current frame index must be passed. This makes the value of the property dependent on the current time.

* It is either a number, string, or boolean
* It uses a DataChannel to store values over time
* It can have an initial static value for when no frame data has been set
* It will return a dynamic value for a given frame index

## DataChannel

A DataChannel is used to store the data for an AnimatableProperty. Data is stored in DataBlocks

* It will return the current value for a given frame by using the DataBlocks
* It uses the DataBlocks start and end values to quickly determine the value of a frame in between DataBlocks

## DataBlock

A DataBlock is a set of animation frame values from a recording session. By grouping the relevant data together, a DataBlock can be shifted easily by updating the startFrame property and making all contained data relative to that. It also makes knowing the start and end values easy.

* It has a value for every frame matching the Project fps
* It has a startFrame property and all data within the DataBlock is relative to the startFrame
* It only stores a subset of the project’s total animation data

## Asset

An Asset is either a static media resource or a defined Puppet. Assets are linked to the file system sources.

* It can be an image, video, audio, or text static resource
* It can be a defined Puppet

## Puppet

A Puppet is a type of Asset which is defined in a special Scene. Creators can assemble static images or placeholders together with Bones to create a rigged character, or static prop. A Puppet acts like a class or template which can be used in multiple separate instances in Scenes. Every time a Puppet is added to a Scene, a new instance is created. Changes to the Puppet will be reflected in the Scenes.

* Can create a rigged character using images and Bones
* Can create a composition of static images for a prop
* Puppets are added to Scenes creating PuppetSprites
* Changes to the Puppet composition will be reflected in all instances

## Joint

A Joint is a connection between two images for a Puppet.

## Bone

A Bone is a connector between joints for a Puppet. Bones help with animations by giving a skeleton with constraints.

* It can have constraints for rotation

## Sprite

A Sprite is a high-level display object used in Scenes. There can be many different types of Sprites though they all inherit a base level of display functionality and animation capabilities. Sprites can be static for scenery or prop purposes, or they can be dynamic and contain animatable data.

Sprite examples are:

* ImageSprite
* PuppetSprite
* TextSprite
* VideoSprite