Beginners Introduction to Beast 2D Game Development Framework

Table of Content

What is Beast 2D

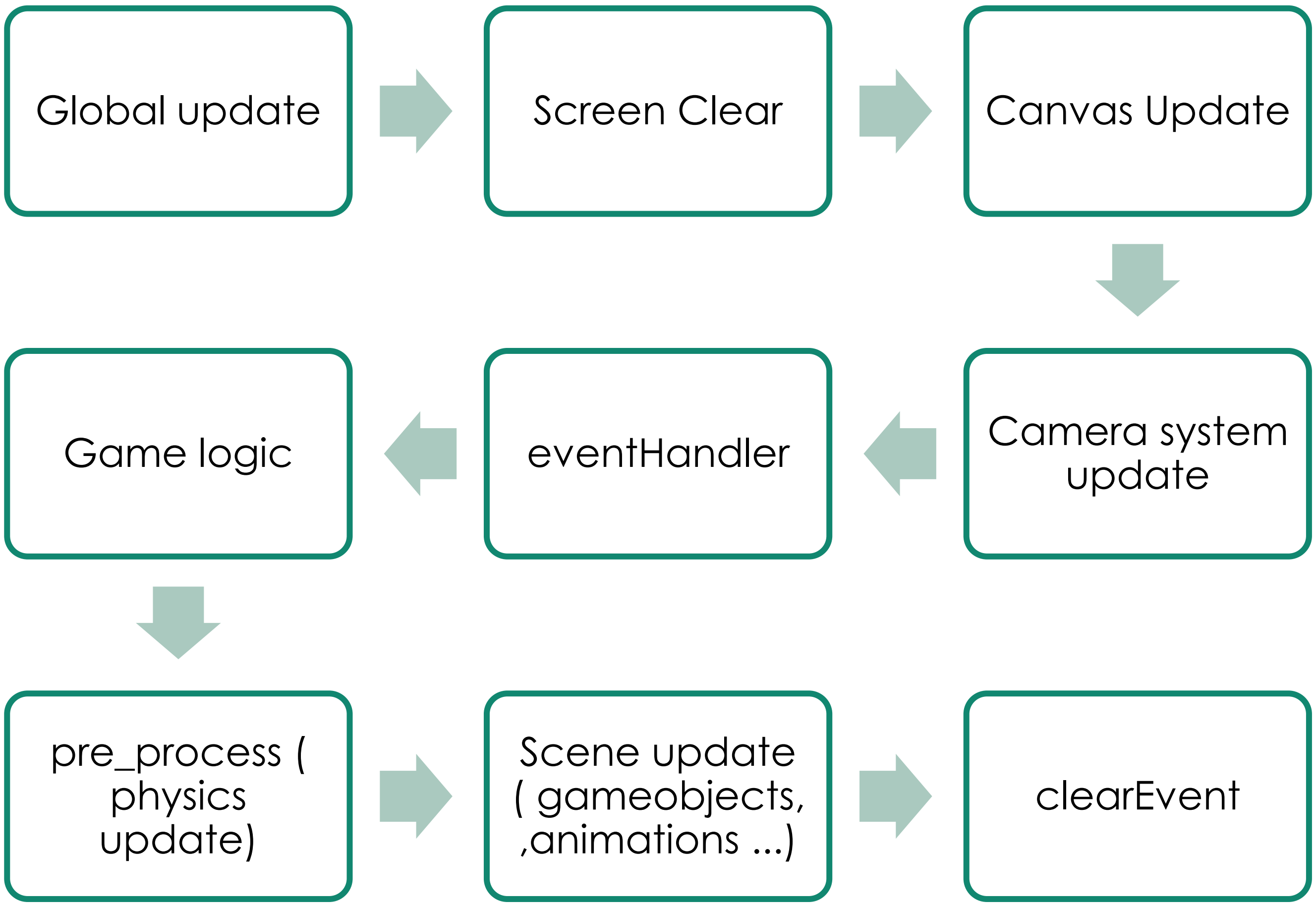
Beast 2D is an open source JavaScript game development framework. Beast 2D provides a variety of features to aid in the creation of 2D games such as sprite animations, graphics and physics system.

Features of Beast 2D

- Free and open source licence
- Sprite animation system
- Event handling system
- Game Entities and component system

Introduction

Beast 2D provides a simple and iterative update structure for game development. The structure is in form of cycle in which the different elements of the game, such as; game objects, backgrounds, animations, physics and events, are updated. The cycle is initiated by a global update function call as shown in the figure below.



The different components of the update cycle can further be described as follows:

Global update: This is a user defined function that determines when the game should start. It implicitly calls the rest of the game component updates.

Screen clear: This function clears the canvas with a user defined color or image each cycle to ensure that the screen display is refreshed at the beginning of each frame.

Canvas update: This is a method of canvas object created by automatically by the system. This function calls the update of the actual elements of the game.

Game logic: This is method of the canvas object which can be changed by to define the logic of the game such as AI and player movement.

Event Handler: This is method that handles input events and signals within in the game. It calls functions that listen to keyboard and mouse objects.

Camera update: This is function that updates the position of the camera and viewport of the game. It also sets offset vectors that define the mouse position in the game world.

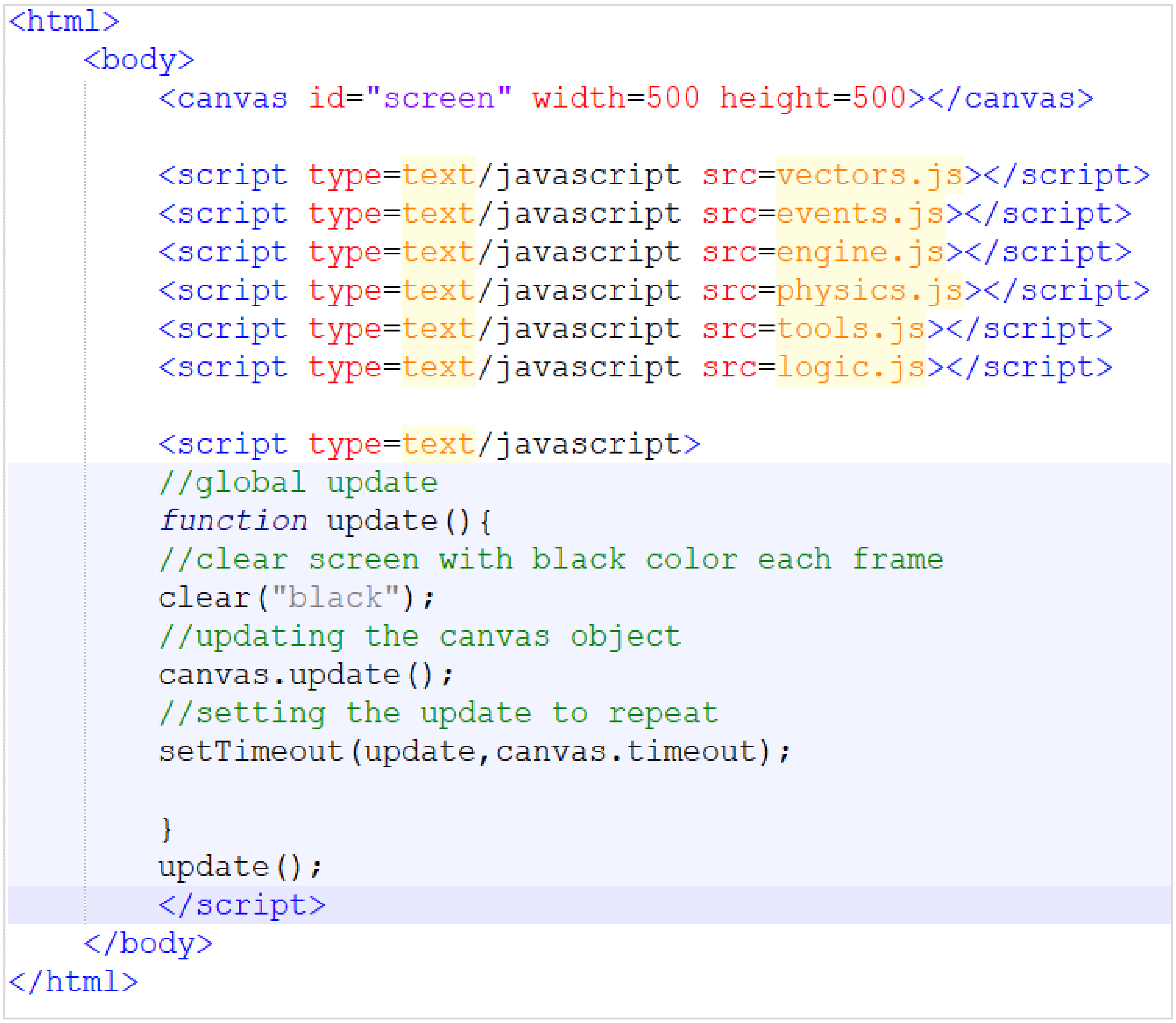
Preprocess: This is a special function that calls the physics system. It handles calls to the rigid body system and collision system.

Scene update: This is a method of the current scene of the game. It calls an update of other elements of the game such as game objects and backgrounds. Main elements of the game are render during this update.

Clear Event: This function clears states of signals of input events in the keyboard and mouse objects.

Startup

To start up your game using Beast 2D, you need to setup the game's update cycle. To use Beast 2D, you need to specify an html canvas element where the game is displayed. This element is given a special id called "screen" as shown in the code below.



```
<html>
    <body>
        <canvas id="screen" width=500 height=500></canvas>

        <script type=text/javascript src=vectors.js></script>
        <script type=text/javascript src=events.js></script>
        <script type=text/javascript src=engine.js></script>
        <script type=text/javascript src=physics.js></script>
        <script type=text/javascript src=tools.js></script>
        <script type=text/javascript src=logic.js></script>

        <script type=text/javascript>
        //global update
        function update(){
        //clear screen with black color each frame
        clear("black");
        //updating the canvas object
        canvas.update();
        //setting the update to repeat
        setTimeout(update,canvas.timeout);

        }
        update();
        </script>
    </body>
</html>
```

From the above code, the game is set to repeat its cycle by timeout value which is an attribute of the canvas object. This value can be changed to improve frame rate of the game.

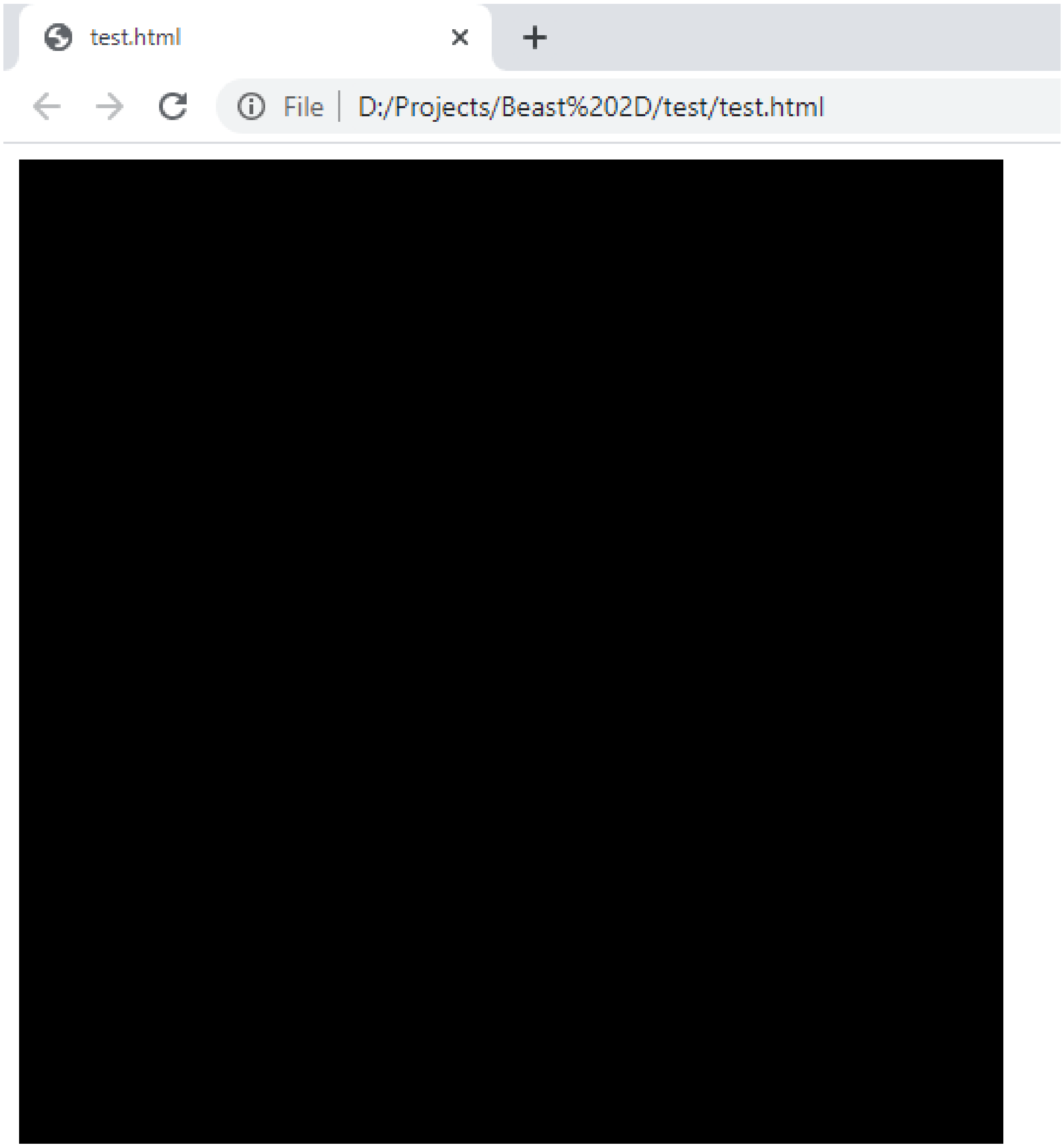


Figure 1 Output of startup code in Chrome browser

Canvas

The canvas is the main object of the game it contains information about all game elements and links them through an update method. The canvas object is automatically created upon including the "engine.js" script. The table describes some the important attributes of the canvas object.

Attribute	**Type**	**Description**
id	String	This is an id of the html canvas element for which the canvas is created. This is id is by default set to "screen" and hence for the game to run, an html canvas element with same id is required.
gl	object	This is object representing the web-gl 2D context object. It is used to draw the canvas graphics.
width	float	This is the width of the viewport of the game.
height	float	This is the height of the viewport of the game.
element	object	This is an object representing the html canvas element onto which the game is drawn.
currentScene	object	This is the scene object which represents the current scene of the game. When this attribute is set, the scene represent by that object is rendered on screen.
timeout	float	This is the timeout value used in the global update cycle. It defines the rate at which the game updates and the frame rate.
globalDelay	float	This is delay value which is used to delay animations by default which that they play smoothly.
scenes	object	This is a dictionary object that stores all scenes created for the game.
initialitze()	method	This is a method that initializes the canvas object and sets its main attributes such as the canvas element.
resetViewport(width,height)	method	This is method that sets the height and width of the viewport. It's called

Attribute	Type	Description
		automatically whenever the viewport is resized. **width**: This is a float defining width of viewport. **height**: This is a float defining the height of the viewport.
pre_process()	method	This a method that calls the physics update. It checks collision of all colliders and movement of rigid bodies.
logic()	method	This is a method that is user defined and executes any logic required in the game such as listening for input events and moving objects.
update()	method	This is a method called by the global update and it updates the main objects of the canvas such as the scenes and physics objects.
addScene(scene)	method	This a method used to add a scene in to the canvas scene dictionary. Every scene added to the canvas using this method can be access through following code: **canvas.scenes[“name of scene”]**
worldCord(x,y)	method	This a method that that executes world transforms on mouse input coordinates if required.

Scene

The scene is a collection of entities with in the game such as ui buttons, graphics, game objects, colliders and rigid bodies. The scene object acts as easy way to create levels and menus for the game.

Attribute	**Type**	**Description**
name	String	This is the name of scene such as "level 1".
backgrounds	Array	This is an array of backgrounds objects with in the scene.
objects	Array	This is an array of game objects and tile objects within the scene.
layouts	Array	This is an array of layout objects with in the scene
polys	Array	This is an array of graphical objects such as lines, circles and rectangles within the scene.
guiObjects	Array	This is an array of ui objects such as ui buttons and images.
colliders	Array	This is an array of physics colliders within the scene.
visuals	Array	This is an array of visual objects such as selector tool.
initialitze()	method	This is a method that initializes the scene. This method makes the scene the current scene if the current scene is not set.
logic()	method	This is a method that is user defined and executes any logic required in the game such as listening for input events and moving objects.
update()	method	This is a method called by the update method of the canvas and it updates the objects of the scene stored in the different arrays.
connect()	method	This method connects the entities of the scene to the

Attribute	Type	Description
		scene by defining which scene they belong to.
reset()	method	This is user defined method that defines state of the objects in the scene when it initializes.
addTile(pos,count,img,collide)	method	This method creates and adds a tile object to the scene. **pos**: This position of the tile as a vector. **count**:This is the number of tiles to be created as an integer. **img:**This is the image object whose data is displayed by the tiles. **collide:** This is a Boolean that determines whether the tile has a collider. If true the tile object is set to have a collider.
addLayout(pos)	method	This method creates and adds layout object to the scene. **pos**: This is the position of the layout on canvas screen as a vector.
addBg(source,type)	method	This method creates and adds background object to the scene. **source**: This is the image object whose data is displayed by the background object. **type**: This determines whether the background object displays an image or a color. If set to "color", a colored background is displayed. If set to "image", an image background is displayed.

Attribute	**Type**	**Description**
addVisual(visual)	method	This method creates and adds a visual object to the scene. **visual**: This is a visual object
drawLine(vec1, vec2,color,width)	method	This method creates and adds a line object to the scene. **vec1**: This is the position of the first point on the line as a vector. **vec2**: This is the position of second point on the line as a vector. **color**: This is the color of the line as a string. **width**: This is the width of the line.
drawRectangle(vec,width,height,color,fill)	method	This method creates and adds a rectangle polygon object to the scene. **vec**: This is the position of the center of the rectangle. **width**: This is the width of the rectangle. **height**: This is the height of the rectangle. **color**: This is the color of the rectangle. **fill:** This is Boolean that determines whether the rectangle is filled with color. If set to true, the rectangle is filled with color.
drawPoint(vec,radius,color,fill)	method	This method creates and adds a point or circle graphics object to the scene. **vec**: This is position of the point object as a vector. **radiu**s: This is the radius of the point drawn. **color**: This is the color of the point drawn.

Attribute	Type	Description
		fill: This is Boolean that determines whether the circle is filled with color. If set to true, the circle is filled with color.
addProgress(val,pos)	method	This method creates and adds progress bar to the scene. **val**: This is the current value of the progress bar. **pos**: This is the position of the progress bar as a vector.
addImageGui(source,pos)	method	This method creates and adds ui image object to the scene. **source**: This is the image object whose data is displayed. **pos**: This is the position of the top left corner of the ui image.
addText(text,pos)	method	This method creates and adds a ui text object to the scene. **text**: This the text displayed the ui text object. **pos**: This is the position of the text object as a vector.
addButton(text,pos)	method	This method creates and adds a ui button object to the scene. **text**: This is the text displayed by the button. **pos**: This is the position of the top left corner of button.
addGameObject(name)	method	This method creates and adds a game object to the scene. **name**: This is the name of the game object to be created.

Attribute	Type	Description
run()	method	This method is runs the scene by making it the current scene and initializing it.

Assets

To ease loading of assets such as audios and images for the game, beast2d uses an asset collection system to load assets. In this system, all links to images and audios for the game have to be defined and a dictionary of image and audio objects is created for use in the game. The asset collection system enables the use of loading screens and progress bars for start of the game. The code below shows how to use the asset collection system.

```
<html>
    <body>
        <canvas id="screen" width=500 height=500></canvas>

        <script type=text/javascript src=vectors.js></script>
        <script type=text/javascript src=events.js></script>
        <script type=text/javascript src=engine.js></script>
        <script type=text/javascript src=physics.js></script>
        <script type=text/javascript src=tools.js></script>
        <script type=text/javascript src=logic.js></script>

        <script type=text/javascript>
        //array of links to images for game
        images=["tile.png","walk.png","idle.png"];
        //array of links to audios for the game
        sounds=["shoot.mp3"];
        //creation of asset dictionary
        init_assets();

        //accessing image objects from asset dictionary
        walkImage= imageSet["walk.png"];
        //accessing audio objects from asset dictionary
        shootSound= soundSet["shoot.mp3"];
```

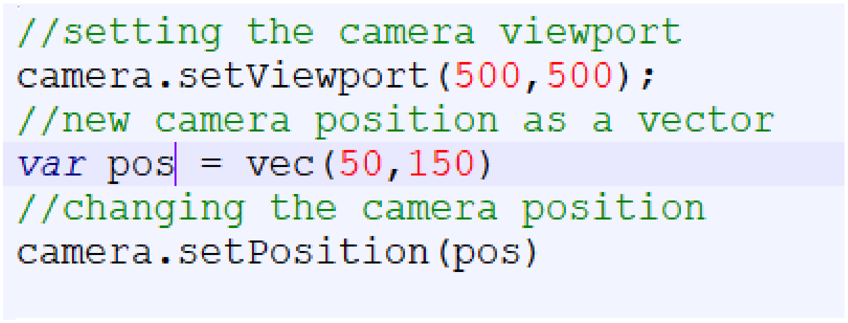
Assets can be loaded at different stages of the game for different scenes or levels as required. Assets that are not needed can also be replaced in memory to create space for those assets you need in game. The images and audio objects from the asset dictionary are the html image objects.

Camera

This is an object represents the viewport of the game. This object's world position can be adjusted to move the camera with in the scene.

Attribute	**Type**	**Description**
position	vec2d object	This is the position of the camera in the game world.
setPosition(vec)	method	This method sets the position of the camera in game. **vec:** This is the position of the camera as a 2d vector.
setViewport(width,height)	method	This method that defines the width and height of the game viewport. **width:** This the width of the viewport **height:** This is the height of the viewport.

The code below shows how to set the viewport and move the camera position in the game



```
//setting the camera viewport
camera.setViewport(500,500);
//new camera position as a vector
var pos = vec(50,150)
//changing the camera position
camera.setPosition(pos)
```

Backgrounds

These are objects that create the background of the game. They can be fill the background with a given color or specified image. Backgrounds are drawn to the full size of the canvas.

Attribute	**Type**	**Description**
position	vec2d object	This is position of the background in the game. This is by default set as the origin.
type	String	This determines whether the background object displays an image or a color. If set to "color", a colored background is displayed. If set to "image", an image background is displayed.
source	Html Image Object	This is the image object whose data is displayed by the background.
delete()	method	This method deletes the background object from the game.

UI features

Beast2d provides a number of features to ease the creation of ui objects such as buttons, labels and images. These features are discussed below:

UI Buttons

UI buttons are used to create menu and interface buttons.

Attribute	**Type**	**Description**
position	vec2d object	This is position of the button on the screen. This is by default set as the null.
rotation	Float	This is the rotation of the button on the screen. This is by default set to zero.
label	UI label object	This is a ui label object that display the text for the button.
tcolor	String	This is the color of the button text as string such as “red”, ”green” .e.t.c
tcolor2	String	This is the color of the button text as string when the cursor is over it.
color	String	This is the background color of the button.
color2	String	This is the background color of the button when the cursor is over it.
scene	Scene object	This is the scene object that represents the scene to which the button belongs.
padding	Float	This is the padding the text with the button.
image1	Html Image Object	This is an image object used to draw the background of the button if defined.
image2	Html Image Object	This is an image object used to draw the background of the button when the cursor is over the button.
id	String	This is the string that identifies the type of ui object that the button belongs to.
delete()	method	This method deletes the button object from the game.
getSize()	method	This method is used to get the size of the button on the screen.
update()	method	This is a method called by the update method of the canvas and it draws the button.
action()	method	This is a user defined method that defines what is done when the button is clicked.

Attribute	Type	Description
setImages(img1,img2)	method	This method that defines the images used to draw the button's background. **img1**: This is the html image object that is used to define the image1 attribute of the button. **img2**: This is the html image object that is used to define the image2 attribute of the button.

UI Label

This feature creates text labels for menus and interfaces.

Attribute	Type	Description
position	vec2d object	This is position of the label on the screen. This is by default set as the canvas center.
rotation	Float	This is the rotation of the label on the screen. This is by default set to zero.
color	String	This is the color of the label text as string such as "red", "green" .e.t.c
scene	Scene object	This is the scene object that represents the scene to which the label belongs.
text	String	This is text displayed by the label.
font	String	This is font the label text such as "Arial".
size	Float	This is the size of the text displayed by the label.
type	String	This attribute sets whether the text drawn is either stroked or filled. The attribute can be set to either "stroke" or "fill".
id	String	This is the string that identifies the type of ui object that the label belongs to.
delete()	method	This method deletes the label object from the game.
getSize()	method	This method is used to get the size of the text displayed the label.
getFont()	method	This method returns the font and size of the text displayed by the label.
setStyle(font,size,type)	method	This method sets the styling parameters of the text displayed by the label.

Attribute	Type	Description
		font: This argument is used to set the font of the label text. **size**: This argument is used to set the text size of the label. **type:** This argument is used to set the type attribute of the label.
update()	method	This is a method called by the update method of the canvas and it draws the label.

UI Image

This feature creates an image for menus and interfaces.

Attribute	Type	Description
position	vec2d object	This is position of the ui image on the screen. This is by default set as the null.
rotation	Float	This is the rotation of the ui image on the screen. This is by default set to zero.
width	Float	This is the width of the ui image.
height	Float	This is the height of the ui image.
scene	Scene object	This is the scene object that represents the scene to which the ui image belongs.
click	Boolean	This is a Boolean value that determines whether the ui image triggers an action when clicked.
image1	Html Image Object	This is an image object used to draw the ui image.
image2	Html Image Object	This is an image object used to draw the ui image when the cursor is over it.
id	String	This is the string that identifies the type of ui object that the ui image belongs to.
delete()	method	This method deletes the ui image object from the game.
getSize()	method	This method is used to get the size of the ui image on the screen.
update()	method	This is a method called by the update method of the canvas and it draws the ui image.

Attribute	Type	Description
action()	method	This is a user defined method that defines what is done when the ui image is clicked.
setImages(img1,img2)	method	This method that defines the images used to draw ui image. **img1**: This is the html image object that is used to define the image1 attribute of the ui image object. **img2**: This is the html image object that is used to define the image2 attribute of the ui image object.

Layouts

This features arranges and spaces ui objects with a menu or interface.

Attribute	Type	Description
position	vec2d object	This is position of the layout on the screen. This is by default set as the zero vector.
spacing	Float	This is the spacing of the objects with in the layout. This is by default set to 30 units
width	Float	This is the width of the layout.
height	Float	This is the height of the layout.
scene	Scene object	This is the scene object that represents the scene to which the layout belongs.
objects	Array	This is an array of objects contained in the layout.
delete()	method	This method deletes the layout from the game.
setMargins(top,bottom,left,right)	method	This method is used to set the margins of the layout. **top:** This is the top margin of the layout. **bottom:** This is the bottom margin of the layout. **left:** This is the left margin of the layout.

Attribute	Type	Description
		right: This is the right margin of the layout.
addObject(object)	method	This is a method adds a ui object to the layout.

Cursor

This features creates a cursor object on the screen with a specified texture.

Attribute	Type	Description
source	html image object	This is an image object used to draw the cursor texture.
visible	Boolean	This is a boolean value that determines whether cursor is visible or not. This is by default set to false.
width	Float	This is the width of the cursor's bounding box.
height	Float	This is the height of the cursor's bounding box.
setImage(image)	method	This method sets the image source of the cursor object.

Game Objects

A game object is the basic object of the game. This can be used to create different elements of the game such as players, enemies, vehicles and weapons. Game objects can have different properties in terms of; physics, materials and animations.

Attribute	**Type**	**Description**
name	String	This is the name of the game object. e.g. player.
position	vec2d object	This is position of the game object in the game.
rotation	Float	This is the rotation of the game object. This is by default set to zero.
components	Array	This is an array of components of the game object such as animations.
materials	Array	This is an array of materials of the game object.
setPosition(vec)	method	This method sets the position of the game object.
addAnimator()	method	This method adds animation object to the game object.
rigidbody()	method	This method returns the rigidbody component of the game object if it exists.
collider()	method	This method returns the collider component of the game object if set.
addComponent(obj)	method	This methods add component object to the game object.
logic()	method	This is a user defined method that is run before the game object is rendered.
update()	method	This method updates the components the game object and draws the game object.

Materials

This a material describes how a game object is drawn.

Attribute	**Type**	**Description**
name	String	This is the name of the material.
object	Game Object	This is the game object to which the material belongs.
width	Float	This is width of the material image rendered.
height	Float	This is height of the material image rendered.
renderer	rectangle object	This is a rectangle object used to draw the material image on the screen.
setSize(w, h)	method	This method is used to set the size of the material image.
update()	method	This is a method is used to draw the material.

Animations

The animation object is used to create and play animations on materials attached to game objects.

Attribute	**Type**	**Description**
Play()	method	This method plays the animation.
AddClip(name,source,w,h,w1,h1)	method	This method adds a sprite animation clip to the animation object. **name:** This is the name of the animation. **source:** This is the sprite image source. **w:** This is the width of the source image. **h:** This is the height of the source image. **w1:** This is the width of the sprite animation frame. **h2:** This is the height of the image animation frame.
update()	method	This is a method is used to update the animation.

Physics:

Beast2D supports a physics system that provides both rigid bodies and colliders. The physics system also supports dynamic collision.

Rigid body

The rigid body component enables the addition of forces and velocity to game objects.

Attribute	**Type**	**Description**
mass	Float	This variable sets the mass of the rigid body.
acceleration	vec2d object	This variable the acceleration of the game Object as a vector.
velocity	vec2d object	This variable sets the velocity of the game Object as a vector.
drag	Float	This variable sets the drag constant of the object's motion.
gravity	Boolean	When set to true, it enables the game Object to be affected by gravity.
active	Boolean	When set to true, it activates the rigid body component on the game Object.
renderer	rectangle object	This is a rectangle object used to draw the material image on the screen.
applyForce(force)	method	This is a method is used to adds force to the game Object. **force:** This vec2d object representing the force to be added to the object.
delete()	method	This method deletes the rigid body component from the game Object.
setVelocity(vel)	method	This method set the velocity of the object. **vel:** This is a vec2d object representing the velocity to be set.

Colliders

Collider component enables the collision detect on the game object.

Attribute	**Type**	**Description**
visualize	Boolean	When set to true, the outline of the collider is made visible.
width	Float	This variable sets the width of a box collider.

Attribute	**Type**	**Description**
height	Float	This variable sets height of a box collider.
radius	Float	This variable sets radius of a spherical collider.
bounce	Float	This variables set how much bounce the collider gets on collision.
color	String	This variable sets the visual color of the collider outline.
static	Boolean	When set to true, the collider remains static on collision with a moving object.
delete()	method	This method deletes the collider component.
trigger(col)	method	This method deletes the rigid body component from the game Object.
getNormals()	method	This method gets the normal of a box collider as vectors.
getBounds()	method	This method gets the dimensions of the collider.
pos()	method	This is method get the position of the collider as a vector.