FzxNGN Documentation

June 17,2023

Overview: fzxNGN is a 2D physics engine library that was ported to QB64 from the Impulse engine written by Randy Gaul https://github.com/RandyGaul/ImpulseEngine.

Features:

- Rigid body simulation
- Circle and polygon primitives
- Joint simulation
- Camera Library
- Input Library
- Finite State Machine helper functions
- Perlin noise library functions
- XML parsing (WIP)
- LERP functions
- FPS helper functions
- Tons of vector and matrix math functions.

A Bare Bone Implementation:

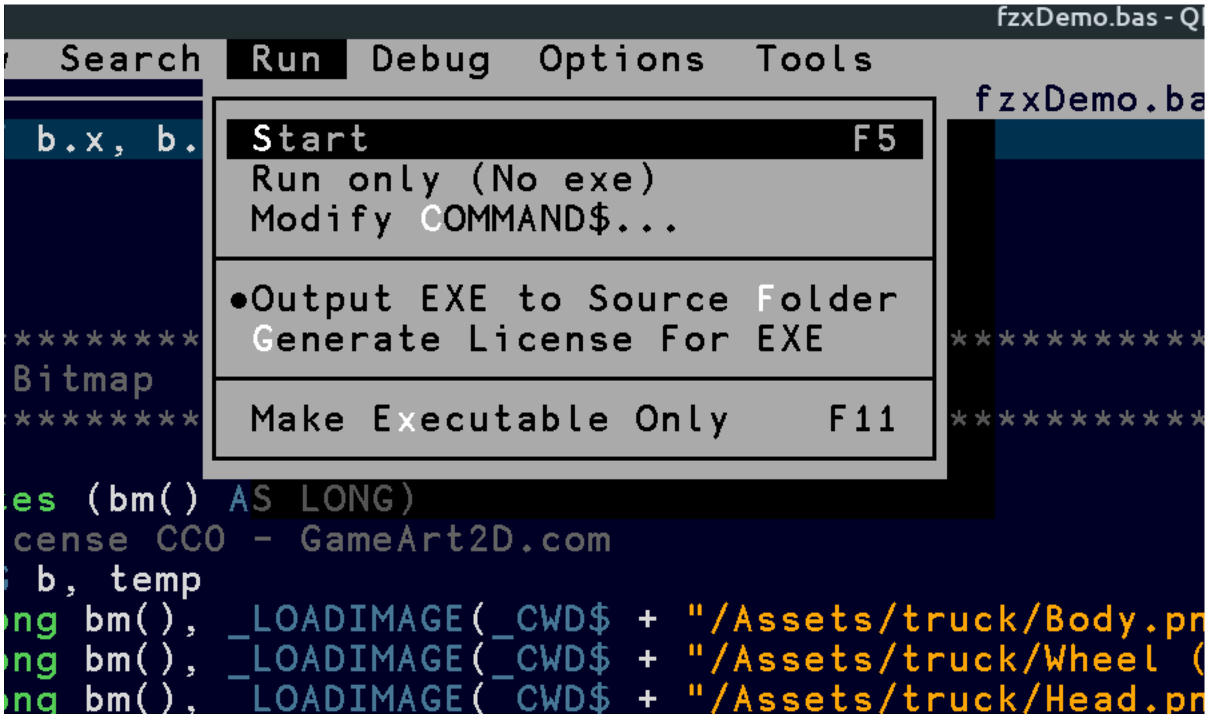
This is a simple example of what it takes for a small implementation of the engine.

- **Initialization include**
  - '$include:'..\fzxNGN_BASE_v2\fzxNGN_ini.bas'
  - sets up the types(UDT), global variables, and constants
- Call to **Build scene.**
- **Main Loop**
  - Call to **Animate scene**
    - Player interaction happens here
  - Call to fzxNGN to calculate the next step
    - **fzxImpulseStep delta time, iterations**
      - Delta Time is your time step i.e. 1/30 of a second
      - Iterations how many steps the simulation runs per step.
  - Call to **Render scene**

- **Include core code**
  - '$include:'..\fzxNGN_BASE_v2\fzxNGN_BASE.bas'
  - All the core functionality is contained here
- **Build Scene**
  - Initial setup of the camera.
  - Set some limits on the world.
  - Setup gravity.
  - Add your bodies to the simulation.
- **Animate Scene**
  - This where you will interact with the bodies.
- **Render Scene**
  - Draw the bodies in the scene.

Compiling examples:

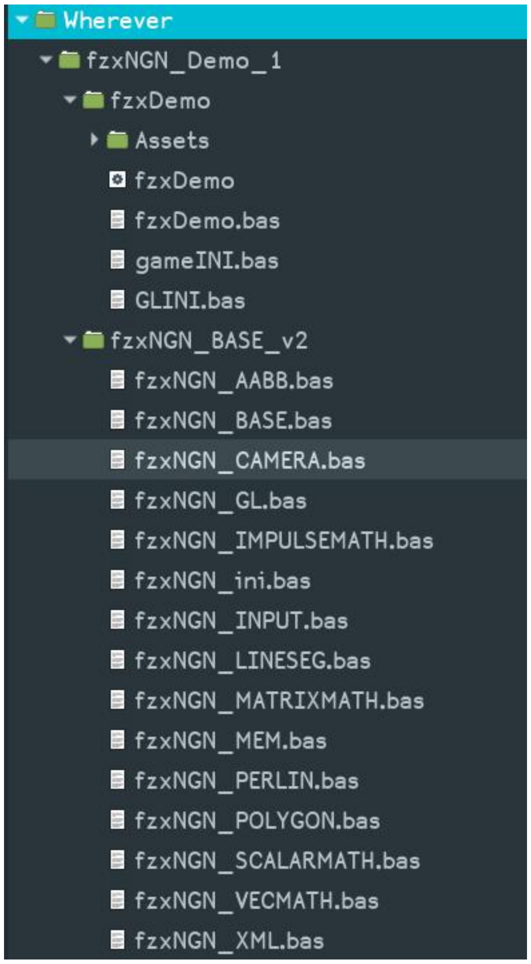
- Compiling should be straightforward however make sure the "Output EXE to Source Folder" is selected. This will ensure that the file structure is not broken.



●Output EXE to Source Folder

```
 b.x, b.
*********                                  ***********
Bitmap
*********                                  ***********
es (bm() AS LONG)
cense CC0 - GameArt2D.com
 b, temp
ng bm(), _LOADIMAGE(_CWD$ + "/Assets/truck/Body.pn
ng bm(), _LOADIMAGE(_CWD$ + "/Assets/truck/Wheel (
ng bm(), _LOADIMAGE(_CWD$ + "/Assets/truck/Head.pn
```

FzxNGN File structure:



```
Wherever
  fzxNGN_Demo_1
    fzxDemo
      Assets
      fzxDemo
      fzxDemo.bas
      gameINI.bas
      GLINI.bas
    fzxNGN_BASE_v2
      fzxNGN_AABB.bas
      fzxNGN_BASE.bas
      fzxNGN_CAMERA.bas
      fzxNGN_GL.bas
      fzxNGN_IMPULSEMATH.bas
      fzxNGN_ini.bas
      fzxNGN_INPUT.bas
      fzxNGN_LINESEG.bas
      fzxNGN_MATRIXMATH.bas
      fzxNGN_MEM.bas
      fzxNGN_PERLIN.bas
      fzxNGN_POLYGON.bas
      fzxNGN_SCALARMATH.bas
      fzxNGN_VECMATH.bas
      fzxNGN_XML.bas
```

FzxNGN Globals (fzxNGN_ini.bas):

__fzxBody(): Contains all the data pertaining to each rigid body.

__fzxJoints(): Contains all the data pertaining to the joints.

__fzxHits(): Collision information

__fzxCamera : Camera data

__fzxWorld : World data

__fzxFPSCount : FPS counting

__fzxInputDevice : Mouse and keyboard

__fzxSettings : Generalized settings. (Currently only mouse double click timing)

Body Creation:

The following function create bodies for the simulation.

fzxCreateCircleBodyEx ("unique name for object", radius)

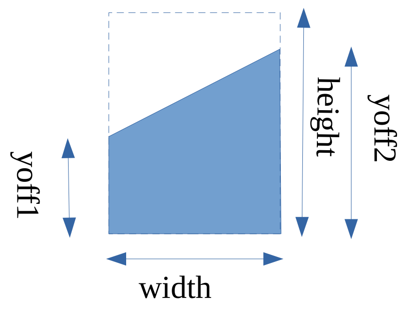
- Adds a circle body to simulation
- returns an index to the body in the __fzxBody() array

fzxCreateBoxBodyEx ("unique name for object", Width, Height)

- Adds a Rectangle to the simulation
- returns an index to the body in the __fzxBody() array

fzxCreateTrapBodyEx ("unique name for object", Width, Height, yoff1, yoff2)

- Adds a trapezoid to the simulation
- returns an index to the body in the __fzxBody() array



An example of a body creation would be like:

1. box = **fzxCreateBoxBodyEx**("box", 100, 100)
2. **fzxSetBody** cFZX_PARAMETER_POSITION, box, 100, 100
3. **fzxSetBody** cFZX_PARAMETER_STATIC, box, 0, 0
4. **fzxSetBodyEx** cFZX_PARAMETER_ORIENT, "box", _D2R(90), 0

Line 1 creates a box named "box" that is 100 units long by 100 units wide. As stated earlier units are arbitrary, so it can be 100 miles or 100 millimeters. Its up to the user to decide.

Line 2 the body is moved to a position of 100, 100. Again units are arbitrary.

Line 3 the body is set to static, and acts as a wall or a solid obstacle. You can still move it or arrange it as you see fit.

Line 4 the body is now addressed by its name instead of index and the orientation is set to 90 degrees.

Body Parameters:

The following parameters can be set by the fzxSetBody subroutine.

- fzxSetBody (Parameter, Index, argument 1, argument 2)
  - Index in the body in the __fzxBody() array you are changing
  - The arguments are the new values.
    - Argument 2 may not always be necessary. Just leave it 0.
  - Parameter Contants
    - cFZX_PARAMETER_POSITION
      - Argument 1 – X position in the world
      - Argument 2 – Y position in the world
    - cFZX_PARAMETER_VELOCITY
      - Argument 1 – X velocity in the world
      - Argument 2 – Y velocity in the world
    - cFZX_PARAMETER_FORCE
      - Argument 1 – X force applied to body
      - Argument 2 – Y force applied to body
    - cFZX_PARAMETER_ANGULARVELOCITY
      - Argument 1 – angular velocity in the world
      - Argument 2 – not used
    - cFZX_PARAMETER_TORQUE
      - Argument 1 – torque force applied to body
      - Argument 2 – not used
    - cFZX_PARAMETER_ORIENT
      - Argument 1 – body angle in radians
      - Argument 2 – not used
    - cFZX_PARAMETER_STATICFRICTION
      - Argument 1 – static friction on the body surface
      - Argument 2 – not used
      - More info https://en.wikipedia.org/wiki/Friction
    - cFZX_PARAMETER_DYNAMICFRICTION
      - Argument 1 – dynamic/kinetic friction on the body surface
      - Argument 2 – not used
      - More info https://en.wikipedia.org/wiki/Friction
    - cFZX_PARAMETER_RESTITUTION
      - Argument 1 – bounciness of the body surface
      - Argument 2 – not used
    - cFZX_PARAMETER_COLOR
      - Argument 1 – color used in wire frame, depends on renderer to implement.
      - Argument 2 – not used

- cFZX_PARAMETER_ENABLE
  - Argument 1 – 0 or non zero
    - Removes body from simulation
    - can be reenabled
  - Argument 2 – not used
- cFZX_PARAMETER_STATIC
  - Sets the object as static and object act like a wall or permanant fixture
  - Argument 1 – not used
  - Argument 2 – not used
- cFZX_PARAMETER_TEXTURE
  - Sets the texture for the body, depends on renderer to implement.
  - Argument 1 – valid texture handle.
  - Argument 2 – not used
- cFZX_PARAMETER_FLIPTEXTURE
  - Flip texture flag, depends on renderer to implement.
  - Argument 1 – 0 or non zero
  - Argument 2 – not used
- cFZX_PARAMETER_SCALETEXTURE
  - Scale texture multiplier, depends on renderer to implement.
  - Argument 1 – X axis, positive non zero number
  - Argument 2 – Y axis, positive non zero number
- cFZX_PARAMETER_OFFSETTEXTURE
  - Shift texture by offset, depends on renderer to implement.
  - Argument 1 – X axis
  - Argument 2 – Y axis
- cFZX_PARAMETER_COLLISIONMASK
  - Used to selectively allow collisions between bodies
    - A value of &B00000001 on one body and value &B00000001 on another body will collide.
    - A value of &B00000010 on one body and &B00000001 on another body will not collide.
    - The default is &B11111111.
    - They essentially logically ANDed together.
  - Argument 1 – unsigned integer
  - Argument 2 – not used
- cFZX_PARAMETER_INVERTNORMALS
  - Experimental feature (I don't recommend using it)
  - Argument 1 – unsigned integer
  - Argument 2 – not used
- cFZX_PARAMETER_NOPHYSICS
  - Used for sensors. Similar to cFZX_PARAMETER_ENABLE, but body still picks up collisions, but wont react to them.
  - Argument 1 – 0 or non zero
  - Argument 2 – not used

- cFZX_PARAMETER_SPECIALFUNCTION
    - User functionality, can be used for whatever the user needs
    - Argument 1 – any value
    - Argument 2 – any value
- cFZX_PARAMETER_RENDERORDER
    - Depreciated – left for compatibility
    - Argument 1 – any value
    - Argument 2 – unused
- cFZX_PARAMETER_ENTITYID
    - User functionality, can be used for whatever the user needs
    - Argument 1 – any value
    - Argument 2 – unused
- cFZX_PARAMETER_LIFETIME
    - Give the body a finite lifetime
    - Argument 1 – time in seconds
    - Argument 2 – unused
- cFZX_PARAMETER_REPEATTEXTURE
    - Repeat texture multiplier, depends on renderer to implement.
    - Argument 1 – X axis, positive non zero number
    - Argument 2 – Y axis, positive non zero number
- cFZX_PARAMETER_ZPOSITION
    - Sets the body render order, depends on renderer to implement.
    - Argument 1 – Z axis
    - Argument 2 – unused
- cFZX_PARAMETER_UV0
    - Texture Coordinates, depends on renderer to implement.
    - Argument 1 – X axis, positive non zero number
    - Argument 2 – Y axis, positive non zero number
- cFZX_PARAMETER_UV1
    - Texture Coordinates, depends on renderer to implement.
    - Argument 1 – X axis, positive non zero number
    - Argument 2 – Y axis, positive non zero number
- cFZX_PARAMETER_UV2
    - Texture Coordinates, depends on renderer to implement.
    - Argument 1 – X axis, positive non zero number
    - Argument 2 – Y axis, positive non zero number
- cFZX_PARAMETER_UV3
    - Texture Coordinates, depends on renderer to implement.
    - Argument 1 – X axis, positive non zero number
    - Argument 2 – Y axis, positive non zero number

Querying Body Parameters:

Making this easier is on the To-Do list. All of the parameters that have been set can be read by looking at the __fzxBody() structure. The structure is defined in the fzxNGN_ini.bas file.

From the earlier example we can look at the current position

1. PRINT __fzxBody(box).fzx.position.x
2. PRINT __fzxBody(box).fzx.position.y