GXPEngine Cheat Sheet

This is an overview of the most important classes, and their public fields (variables and get-/setters) and methods. Feel free to extend this document for yourself!

Table of Contents

Game Object Classes

Transformable

Contains the position, rotation and scale of game objects (relative to the current parent), and methods to modify those and change between (parent/child) spaces.

Fields:

float x,y	Position in pixels. Origin = top left
float rotation	Rotation in degrees (positive = clockwise)
float scaleX,Y	The x and y scale (default = 1)
float scale	Sets both x and y scale

Methods:

SetXY(x,y)	Sets the position
Turn(angle)	Change the rotation by the given angle in degrees
Move(stepX,stepY)	Move the object, based on its current rotation
Translate(stepX,stepY)	Move the object in its parent space (ignore rotation)
SetScaleXY(scaleX,scaleY)	Sets the scale

GameObject

Inherits from Transformable.

Contains information about the game object hierarchy (parent, children), and methods to add and destroy game objects. A game object is only rendered if one of its ancestors is the current game (= *active* game object). In addition, for those active game objects, the following methods are called if you implement them in a sub class:

void Update()	Called once every frame
void OnCollision(GameObject other)	Called every frame, for every active game object that overlaps with this game object

Fields:

bool visible	Whether the GameObject and its children should be rendered
Game game	Returns a reference to the current Game
GameObject parent	The parent game object

Methods, hierarchy related:

Destroy()	Destroy the game object and all its children
LateDestroy()	Destroys the game object after finishing the current Update and OnCollision loop
AddChild()	Adds another game object as child of this one
LateAddChild()	Adds another game object as child of this one after finishing the current Update and OnCollision loop
GetChildren()	Returns a list of all child objects
FindObjectOfType<T>()	Returns the first game object of type T, found within the descendants of this game object
FindObjectsOfType<T>()	Returns all game objects of type T, found within the descendants of this game object

Methods, collision related:

GetCollisions()	Returns a list of all objects (with colliders) that currently overlap this one
HitTest(other)	Returns whether this object overlaps with an other GameObject (both must have colliders)
HitTestPoint(x, y)	Returns whether a point x, y in screen space overlaps with this game object
MoveUntilCollision(vx, vy)	Moves the object by vx, vy (in parent space), until it collides with another object with a solid collider. If so, returns a Collision object

Sprite

Inherits from GameObject.

Represents an image on screen, and holds information on how the image should be rendered (color, etc).

Fields:

int width	The width of the sprite. This is the width in pixels, multiplied by the current scaleX (rounded)
int height	The height of the sprite. This is the height in pixels, multiplied by the current scaleY (rounded)
Float alpha	The opacity of the sprite. 0=transparent, 1=opaque

Methods:

Sprite(filename)	**Main constructor**. Creates a sprite from an image file (png, jpg, etc.)
SetOrigin(x,y)	Sets the (rotation) origin of the sprite, in pixels. Typical use: SetOrigin(width/2,height/2), *before* scaling the sprite
Mirror(mirrorX, mirrorY)	Enables mirroring (reflecting) the sprite in two directions
SetColor(r,g,b)	Sets the color of the sprite. Pass red, green and blue values between 0 and 1.

AnimationSprite

Inherits from Sprite.

A sprite that supports multiple frames, to enable sprite animation. The image file needs to contain the animation frames in a regular grid, with a certain number of columns and rows.

Fields:

int currentFrame	The current animation frame (typically between 0 and rows*cols-1)

Methods:

AnimationSprite(filename, cols, rows)	**Main constructor**. Creates an animation sprite from an image file containing a grid of frames, with given number of columns and rows
SetFrame(frame)	Sets the current animation frame to be shown. (Typically between 0 and cols*rows-1)
SetCycle(startFrame, numFrames)	Sets the current animation cycle to the range from startFrame to startFrame+numFrames-1. Use this with Animate(). numFrames needs to be at least 1.
Animate(deltaFrameTime)	Animates the sprite using the currently selected cycle of frames. Lower values (than 1) for deltaFrameTime slow down the animation.

EasyDraw

Inherits from Canvas, which inherits from Sprite.

A canvas that can be drawn on, with methods similar to Processing, which use current drawing settings (fill and stroke color and width, font, etc).

For a full overview of its methods, see the file itself.

Constructor:

EasyDraw(width,height)	Creates a canvas with the given (pixel) width and height

Changing drawing settings:

TextFont(newFont)	Sets the font. See also Utils.LoadFont
TextAlign(horizontal,vertical)	Sets the alignment mode for drawing text
ShapeAlign(horizontal,vertical)	Sets the alignment mode for drawing shapes

Stroke(red, green, blue, alpha)	Sets the outline color and transparency for drawing shapes. Use values between 0 and 255.
StrokeWeight(width)	Sets the outline width for drawing shapes
NoStroke()	Disable outline drawing
Fill(red, green, blue, alpha)	Sets the fill color for drawing shapes and text. Use values between 0 and 255.
NoFill()	Disable fill for shape drawing

Drawing (using current settings):

Clear(red, green, blue, alpha)	Clears the canvas with the given color. Use alpha=0 for a transparent clear. Use values between 0 and 255.
Text(text, x, y)	Draws the given text string at position x,y (taking alignment settings into account)
Rect(x,y,width,height)	Draws a rectangle at the given point, with the given width and height
Ellipse(x,y,width,height)	Draws an ellipse at the given point, with the given width and height
Line(x1,y1,x2,y2)	Draws a line from (x1,y1) to (x2,y2)

Game

Inherits from GameObject.

The root game object in the hierarchy. There is always exactly one game. Only game objects that are a descendant of game (="in the hierarchy") are rendered, collision checked, and updated.

By default, the logical width and height (as used by object coordinates and mouse position) is the same as the actual window width and height, but this may be changed, e.g. when resizing the window.

Fields:

width	The (logical) width of the window (get only)
height	The (logical) height of the window (get only)
currentFps	The current frame rate (get only)
targetFps	The target fps, for when VSync is disabled

Methods:

Game (int width, int height, bool fullScreen, bool VSync = true, int realWidth=-1, int realHeight=-1, bool pixelArt=false)	The main constructor. Constructs a game with logical dimensions (width, height), and actual window dimensions (realWidth,realHeight). fullScreen: whether the game should run in fullscreen mode (set to false during development!). VSync: whether the game render rate should match the monitor refresh rate (typically 60Hz). If not, targetFps is used. pixelArt: if false, textures are interpolated, resulting in a smooth look. Use true for pixel art.
GetDiagnostics()	Returns a string with various info about the data (game objects, etc.) in the game. useful for debugging and optimization.

Other Classes

Sound

Represents a sound resource. You can load mp3, ogg or wav files.

Methods:

Sound(filename, looping, streaming)	Default constructor. Loads a sound from file. Set looping=true for looping sounds (e.g. music tracks), and streaming=true for large sound files (e.g. music tracks)
Play()	Play the sound. Returns a SoundChannel that can be used to change the settings (e.g. volume, pan), and to stop the playback.

SoundChannel

Represents a sound channel, that can play a single sound. This holds settings such as volume, panning, and playback rate.

Fields:

float Frequency	The channel frequency. The default is usually 44100 (Hz). Use this to slow down/speed up/pitch shift the sound.
bool Mute	Whether the channel is muted (volume = 0, but may still be playing)

float Pan	The (stereo) pan. Between -1 and 1, where 0=center.
bool IsPaused	Whether the channel is paused (not playing)
bool IsPlaying	Whether the channel is playing (get only)
float Volume	The volume of the channel (between 0 and 1)

Methods:

Stop()	Use this to stop playback

Input

A static class, for getting keyboard and mouse input. Use the Key class for key codes. (E.g. Input.GetKey(Key.SPACE))

Fields:

mouseX	The x coordinate of the mouse, relative to the (logical) window coordinates (get only)
mouseY	The y coordinate of the mouse, relative to the (logical) window coordinates (get only)

Methods:

GetKey(key)	Whether the given key is currently pressed
GetKeyDown(key)	Whether the given key was pressed down this frame
GetKeyUp(key)	Whether the given key was released this frame
AnyKey()	Whether any key is currently pressed
AnyKeyDown()	Whether any key was pressed down this frame
GetMouseButton(button)	Whether the given button is currently pressed (button=0,1,2)
GetMouseButtonDown(button)	Whether the given button was pressed down this frame
GetMouseButtonUp(button)	Whether the given button was released this frame

Time

A static class that gives information about the time since program start, and the delta time between frames.

Fields:

time	Time since the program started, in milliseconds
deltaTime	Time since last frame, in milliseconds

Mathf

A static class that contains float based mathematical functions such as Sin, Cos, Max, Min, Sqrt, Clamp, Sign, Abs, Round, and more. See the class itself for details.

Utils

A static class that contains various utility methods, such as Random and LoadFont. See the class itself for details.