Expressions reference

Expressions can be entered when you see a field with one of these buttons:

- The left button indicates a “string expression” (a text)
- The right button indicates a “numerical expression” (a number)

This page is a reference of all expressions that can be used in GDevelop, grouped by the extension, object or behavior they belong too. When `Object` is written, you should enter an object name. **Learn more here about how to write expressions.**

> Expressions are sometime also called functions, like in mathematics.

Objects

Common features that can be used for all objects in GDevelop. Read more explanations about it.

Expression	Description
`PickedInstancesCount(objectListOrEmptyWithoutPicking)`	Return the number of instances picke by the previous conditions (or actions *objectListOrEmptyWithoutPicking* Obj
`SceneInstancesCount(objectListOrEmptyWithoutPicking)`	Return the number of instances of the specified objects living on the scene. *objectListOrEmptyWithoutPicking* Obj

Expression	Description
`Object.Angle()`	Current angle, in degrees, of the object
`Object.AngleToObject(object)`	Compute the angle between two objects (in degrees). If you need the angle to an arbitrary position, use AngleToPosition. *object* Object
`Object.AngleToPosition(number, number)`	Compute the angle between the object center and a “target” position (in degrees). If you need the angle between two objects, use AngleToObject. *number* Target X position *number* Target Y position
`Object.BoundingBoxBottom()`	Return the bounding box (the area encapsulating the object) bottom position.

Expression	Description
`Object.BoundingBoxCenterX()`	Return the bounding box (the area encapsulating the object) center X position.
`Object.BoundingBoxCenterY()`	Return the bounding box (the area encapsulating the object) center Y position.
`Object.BoundingBoxLeft()`	Return the bounding box (the area encapsulating the object) left position.
`Object.BoundingBoxRight()`	Return the bounding box (the area encapsulating the object) right position.
`Object.BoundingBoxTop()`	Return the bounding box (the area encapsulating the object) top position.
`Object.CenterX()`	Return the X position of the center of rotation.
`Object.CenterY()`	Return the Y position of the center of rotation.
`Object.Distance(object)`	Distance between two objects *object* Object
`Object.DistanceToPosition(number, number)`	Distance between an object and a position *number* Target X position *number* Target Y position
`Object.ForceAngle()`	Angle of the sum of forces (in degrees)
`Object.ForceLength()`	Length of the sum of forces
`Object.ForceX()`	X coordinate of the sum of forces
`Object.ForceY()`	Y coordinate of the sum of forces
`Object.Height()`	Height of the object
`Object.Layer()`	Return the name of the layer the object is on
`Object.ObjectName()`	Return the name of the object
`Object.ObjectTimerElapsedTime(identifier)`	Value of an object timer *identifier* Timer's name
`Object.SqDistance(object)`	Square distance between two objects *object* Object
`Object.SqDistanceToPosition(number, number)`	Square distance between an object and a position *number* Target X position *number* Target Y position
`Object.Variable(objectvar)`	Value of an object variable *objectvar* Variable
`Object.VariableChildCount(objectvar)`	Number of children of an object variable *objectvar* Variable
`Object.VariableString(objectvar)`	Text of an object variable *objectvar* Variable
`Object.Width()`	Width of the object
`Object.X()`	X position of the object
`Object.XFromAngleAndDistance(number, number)`	Compute the X position when given an angle and distance relative to the starting object. This is also known as getting the cartesian coordinates of a 2D vector, using its polar

Expression	Description	
	coordinates.	
	number	Angle, in degrees
	number	Distance
`Object.Y()`	Y position of the object	
`Object.YFromAngleAndDistance(number, number)`	Compute the Y position when given an angle and distance relative to the starting object. This is also known as getting the cartesian coordinates of a 2D vector, using its polar coordinates.	
	number	Angle, in degrees
	number	Distance
`Object.ZOrder()`	Z-order of an object	

Sprite

Animated object which can be used for most elements of a game Read more explanations about it.

Expression	Description
`Object.Animation()`	Animation of the object
`Object.AnimationName()`	Name of the animation of the object
`Object.AnimationSpeedScale()`	Animation speed scale
`Object.Opacity()`	Opacity
`Object.PointX(objectPointName)`	X position of a point *objectPointName* Name of the point
`Object.PointY(objectPointName)`	Y position of a point *objectPointName* Name of the point
`Object.ScaleX()`	Scale of the width of an object
`Object.ScaleY()`	Scale of the height of an object
`Object.Sprite()`	Animation frame of the object

Conversion

Expressions to convert number, texts and quantities. Read more explanations about it.

Expression	Description	
`GlobalVarToJSON(globalvar)`	Convert a global variable to JSON	
	globalvar	The global variable to be stringified
`LargeNumberToString(number)`	Convert the result of the expression to text, without using the scientific notation	
	number	Expression to be converted to text
`ObjectVarToJSON(object, objectvar)`	Convert an object variable to JSON	
	object	The object with the variable

Expression	Description
	objectvar The object variable to be stringified
`ToDeg(number)`	Converts the angle, expressed in radians, into degrees *number* Angle, in radians
`ToJSON(scenevar)`	Convert a scene variable to JSON *scenevar* Scene variable to be stringified
`ToNumber(string)`	Convert the text to a number *string* Text to convert to a number
`ToRad(number)`	Converts the angle, expressed in degrees, into radians *number* Angle, in degrees
`ToString(number)`	Convert the result of the expression to text *number* Expression to be converted to text

Variables

Actions, conditions and expressions to handle variables, from simple variables like the player score, the number of remaining lives to complex variables containing arbitrary data like an inventory or the result of a web request. Read more explanations about it.

Expression	Description
`GlobalVariable(globalvar)`	Value of a global variable *globalvar* Name of the global variable
`GlobalVariableChildCount(globalvar)`	Number of children of a global variable *globalvar* Variable
`GlobalVariableString(globalvar)`	Text of a global variable *globalvar* Variable
`Variable(scenevar)`	Value of a scene variable *scenevar* Variable
`VariableChildCount(scenevar)`	Number of children of a scene variable *scenevar* Variable
`VariableString(scenevar)`	Text of a scene variable *scenevar* Variable

Mouse and touch

Conditions and actions to handle either the mouse or touches on touchscreen. By default, conditions related to the mouse will also handle the touches - so that it's easier to handle both in your game. You can disable this behavior if you want to handle them separately in different events. Read more explanations about it.

Expression	Description
`MouseWheelDelta()`	Mouse wheel displacement
`MouseX(layer, number)`	Return the X position of the cursor or of a touch. *layer* Layer (base layer if empty) *Optional*.

Expression	Description	
	number	Camera number (default : 0) *Optional*.
`MouseY(layer, number)`	Return the Y position of the cursor or of a touch.	
	layer	Layer (base layer if empty) *Optional*.
	number	Camera number (default : 0) *Optional*.
`StartedTouchCount()`	The number of touches that have just started on this frame. The touch identifiers can be accessed using StartedTouchId().	
`StartedTouchId(number)`	The identifier of the touch that has just started on this frame. The touch number of touches can be accessed using StartedTouchCount().	
	number	Touch index
`TouchX(number, layer, number)`	Return the X position of a specific touch.	
	number	Touch identifier
	layer	Layer (base layer if empty) *Optional*.
	number	Camera number (default : 0) *Optional*.
`TouchY(number, layer, number)`	Return the Y position of a specific touch.	
	number	Touch identifier
	layer	Layer (base layer if empty) *Optional*.
	number	Camera number (default : 0) *Optional*.

Keyboard

Allows your game to respond to keyboard input. Note that this does not work with on-screen keyboard on touch devices: use instead conditions related to touch when making a game for mobile/touchscreen devices. Read more explanations about it.

Expression	Description
`LastPressedKey()`	Get the name of the latest key pressed on the keyboard

Scene

Actions and conditions to manipulate the scenes during the game.

Expression	Description
`CurrentSceneName()`	Name of the current scene

Timers and time

Actions and conditions to run timers, get the current time or modify the time scale (speed at which the game is running - useful for slow motion effects). Read more explanations about it.

Expression	Description
`Time(string)`	Current time

Expression	Description	
	string	Hour: hour - Minutes: min - Seconds: sec - Day of month: mday - Months since January: mon - Year since 1900: year - Days since Sunday: wday - Days since Jan 1st: yday - Timestamp (ms): timestamp“
`TimeDelta()`	Time elapsed since the last frame rendered on screen	
`TimeFromStart()`	Time elapsed since the beginning of the scene	
`TimeScale()`	Returns the time scale of the scene.	
`TimerElapsedTime(identifier)`	Value of a scene timer	
	identifier	Timer's name

Mathematical tools

A set of mathematical functions that can be used in expressions.

Expression	Description	
`AngleBetweenPositions(number, number, number, number)`	Compute the angle between two positions (in degrees).	
	number	First point X position
	number	First point Y position
	number	Second point X position
	number	Second point Y position
`AngleDifference(number, number)`	Difference between two angles	
	number	First angle, in degrees
	number	Second angle, in degrees
`DistanceBetweenPositions(number, number, number, number)`	Compute the distance between two positions.	
	number	First point X position
	number	First point Y position
	number	Second point X position
	number	Second point Y position
`Pi()`	The number Pi (3.1415...)	
`Random(number)`	Random integer	
	number	Maximum value
`RandomFloat(number)`	Random float	
	number	Maximum value
`RandomFloatInRange(number, number)`	Random float in range	
	number	Minimum value
	number	Maximum value
`RandomInRange(number, number)`	Random integer in range	
	number	Minimum value
	number	Maximum value
`RandomWithStep(number, number, number)`	Random value in steps	
	number	Minimum value

Expression	Description	
	number	Maximum value
	number	Step
`XFromAngleAndDistance(number, number)`	Compute the X position when given an angle and distance relative to the origin (0;0). This is also known as getting the cartesian coordinates of a 2D vector, using its polar coordinates.	
	number	Angle, in degrees
	number	Distance
`YFromAngleAndDistance(number, number)`	Compute the Y position when given an angle and distance relative to the origin (0;0). This is also known as getting the cartesian coordinates of a 2D vector, using its polar coordinates.	
	number	Angle, in degrees
	number	Distance
`abs(number)`	Absolute value	
	number	Expression
`acos(number)`	Arccosine, return an angle (in radian). `ToDeg` allows to convert it to degrees.	
	number	Expression
`acosh(number)`	Hyperbolic arccosine	
	number	Expression
`asin(number)`	Arcsine, return an angle (in radian). `ToDeg` allows to convert it to degrees.	
	number	Expression
`asinh(number)`	Arcsine	
	number	Expression
`atan(number)`	Arctangent, return an angle (in radian). `ToDeg` allows to convert it to degrees.	
	number	Expression
`atan2(number, number)`	2 argument arctangent (atan2)	
	number	Y
	number	X
`atanh(number)`	Hyperbolic arctangent	
	number	Expression
`cbrt(number)`	Cube root	
	number	Expression
`ceil(number)`	Round number up to an integer	
	number	Expression
`ceilTo(number, number)`	Round number up to the Nth decimal place	
	number	Expression
	number	Expression *Optional*.
`clamp(number, number, number)`	Restrict a value to a given range	
	number	Value
	number	Min

Expression	Description	
	number	Max
`cos(number)`	Cosine of an angle (in radian). If you want to use degrees, use `ToRad`: `sin(ToRad(45))`.	
	number	Expression
`cosh(number)`	Hyperbolic cosine	
	number	Expression
`cot(number)`	Cotangent of a number	
	number	Expression
`csc(number)`	Cosecant of a number	
	number	Expression
`exp(number)`	Exponential of a number	
	number	Expression
`floor(number)`	Round number down to an integer	
	number	Expression
`floorTo(number, number)`	Round number down to the Nth decimal place	
	number	Expression
	number	Expression *Optional*.
`lerp(number, number, number)`	Linearly interpolate a to b by x	
	number	a (in a+(b-a) * x)
	number	b (in a+(b-a) * x)
	number	x (in a+(b-a) * x)
`log(number)`	Logarithm	
	number	Expression
`log10(number)`	Base-10 logarithm	
	number	Expression
`log2(number)`	Base 2 Logarithm	
	number	Expression
`max(number, number)`	Maximum of two numbers	
	number	First expression
	number	Second expression
`min(number, number)`	Minimum of two numbers	
	number	First expression
	number	Second expression
`mod(number, number)`	x mod y	
	number	x (as in x mod y)
	number	y (as in x mod y)
`normalize(number, number, number)`	Remap a value between 0 and 1.	
	number	Value
	number	Min
	number	Max
`nthroot(number, number)`	Nth root of a number	
	number	Number
	number	N

Expression	Description	
`pow(number, number)`	Raise a number to power n	
	number	Number
	number	The exponent (n in “x to the power n”)
`round(number)`	Round a number	
	number	Expression
`roundTo(number, number)`	Round a number to the Nth decimal place	
	number	Expression
	number	Expression *Optional*.
`sec(number)`	Secant	
	number	Expression
`sign(number)`	Return the sign of a number (1,-1 or 0)	
	number	Expression
`sin(number)`	Sine of an angle (in radian). If you want to use degrees, use`ToRad`: `sin(ToRad(45))`.	
	number	Expression
`sinh(number)`	Hyperbolic sine	
	number	Expression
`sqrt(number)`	Square root of a number	
	number	Expression
`tan(number)`	Tangent of an angle (in radian). If you want to use degrees, use`ToRad`: `tan(ToRad(45))`.	
	number	Expression
`tanh(number)`	Hyperbolic tangent	
	number	Expression
`trunc(number)`	Truncate a number	
	number	Expression

Layers and cameras

Each scene can be composed of multiple layers. These conditions and actions allow to manipulate them during the game. In particular, you can move the camera of a layer to center it on an object or a position. Read more explanations about it.

Expression	Description	
`CameraAngle(layer, number)`	Return the angle of rotation of a camera (in degrees).	
	layer	Layer (base layer if empty) *Optional*.
	number	Camera number (default : 0) *Optional*.
`CameraBorderBottom(layer, number)`	Return the position of the bottom border of a camera.	
	layer	Layer (base layer if empty) *Optional*.
	number	Camera number *Optional*.
`CameraBorderLeft(layer, number)`	Return the position of the left border of a camera.	
	layer	Layer (base layer if empty) *Optional*.

Expression	Description
	number Camera number *Optional*.
`CameraBorderRight(layer, number)`	Return the position of the right border of a camera. *layer* Layer (base layer if empty) *Optional*. *number* Camera number *Optional*.
`CameraBorderTop(layer, number)`	Return the position of the top border of a camera. *layer* Layer (base layer if empty) *Optional*. *number* Camera number *Optional*.
`CameraCenterX(layer, number)`	Return the X position of the center of a camera. *layer* Layer (base layer if empty) *Optional*. *number* Camera number (default : 0) *Optional*.
`CameraCenterY(layer, number)`	Return the Y position of the center of a camera. *layer* Layer (base layer if empty) *Optional*. *number* Camera number (default : 0) *Optional*.
`CameraHeight(layer, number)`	Return the height of a camera of a layer. *layer* Layer (base layer if empty) *Optional*. *number* Camera number *Optional*.
`CameraWidth(layer, number)`	Return the width of a camera of a layer. *layer* Layer (base layer if empty) *Optional*. *number* Camera number *Optional*.
`CameraZoom(layer, number)`	Zoom of a camera of a layer *layer* Layer *Optional*. *number* Camera number (default : 0) *Optional*.
`LayerDefaultZOrder(layer)`	Default Z Order for a layer *layer* Layer
`LayerTimeScale(layer)`	Returns the time scale of the specified layer. *layer* Layer

---

Sounds and music

GDevelop provides several conditions and actions to play audio files. They can be either long music or short sound effects. Read more explanations about it.

Expression	Description
`GlobalVolume()`	Global volume value
`MusicChannelPitch(number)`	Music's pitch *number* Channel
`MusicChannelPlayingOffset(number)`	Music playing offset *number* Channel
`MusicChannelVolume(number)`	Music volume *number* Channel
`SoundChannelPitch(number)`	Sound's pitch *number* Channel
`SoundChannelPlayingOffset(number)`	Sound playing offset

Expression	Description	
SoundChannelVolume(number)	*number*	Channel
	Sound volume	
	number	Channel

Game window and resolution

Provides actions and conditions to manipulate the game window. Depending on the platform on which the game is running, not all of these features can be applied. Read more explanations about it.

Expression	Description
SceneWindowHeight()	Height of the scene window (or scene canvas for HTML5 games)
SceneWindowWidth()	Width of the scene window (or scene canvas for HTML5 games)
ScreenHeight()	Height of the screen (or the page for HTML5 games in browser)
ScreenWidth()	Width of the screen (or the page for HTML5 games in browser)
WindowTitle()	Window's title

Text manipulation

Provides expressions to manipulate strings (also called texts).

Expression	Description	
FromCodePoint(number)	Get character from code point	
	number	Code point
NewLine()	Insert a new line	
StrAt(string, number)	Get a character from a text	
	string	Text
	number	Position of the character (the first letter is at position 0)
StrFind(string, string)	Search in a text (return the position of the result or -1 if not found)	
	string	Text
	string	Text to search for
StrFindFrom(string, string, number)	Search in a text, starting from a position (return the position of the result or -1 if not found)	
	string	Text
	string	Text to search for
	number	Position of the first character in the string to be considered in the search
StrFindLast(string, string)	Search the last occurence in a string (return the position of the result, from the beginning of the string, or -1 if not found)	
	string	Text
	string	Text to search for

Expression	Description	
`StrFindLastFrom(string, string, number)`	Search in a text the last occurence, starting from a position (return the position of the result, from the beginning of the string, or -1 if not found)	
	string	Text
	string	Text to search for
	number	Position of the last character in the string to be considered in the search
`StrLength(string)`	Length of a text	
	string	Text
`StrRepeat(string, number)`	Repeat a text	
	string	Text to repeat
	number	Repetition count
`SubStr(string, number, number)`	Get a portion of a text	
	string	Text
	number	Start position of the portion (the first letter is at position 0)
	number	Length of the portion
`ToLowerCase(string)`	Lowercase a text	
	string	Text
`ToUpperCase(string)`	Uppercase a text	
	string	Text

Event functions

Advanced control features for functions made with events.

Expression	Description	
`GetArgumentAsNumber(functionParameterName)`	Get function parameter (also called “argument”) value.	
	functionParameterName	Parameter name
`GetArgumentAsString(functionParameterName)`	Get function parameter (also called “argument”) text.	
	functionParameterName	Parameter name

Platform (from extension Platform behavior)

Flag objects as being platforms which characters can run on. Read more explanations about it.

No expressions for this behavior.

Platformer character (from extension Platform behavior)

Jump and run on platforms. Read more explanations about it.

Expression	Description
`Object.PlatformerObject::Acceleration()`	Return the horizontal acceleration of the object (in pixels per second per second).
`Object.PlatformerObject::CurrentFallSpeed()`	Return the current fall speed of the object (in pixels per second). Its value is always positive.
`Object.PlatformerObject::CurrentJumpSpeed()`	Current jump speed
`Object.PlatformerObject::CurrentSpeed()`	Return the current horizontal speed of the object (in pixels per second). The object moves to the left with negative values and to the right with positive ones
`Object.PlatformerObject::Deceleration()`	Return the horizontal deceleration of the object (in pixels per second per second).
`Object.PlatformerObject::Gravity()`	Return the gravity applied on the object (in pixels per second per second).
`Object.PlatformerObject::JumpSpeed()`	Return the jump speed of the object (in pixels per second). Its value is always positive.
`Object.PlatformerObject::JumpSustainTime()`	Return the jump sustain time of the object (in seconds).This is the time during which keeping the jump button held allow the initial jump speed to be maintained.
`Object.PlatformerObject::LadderClimbingSpeed()`	Return the ladder climbing speed of the object (in pixels per second).
`Object.PlatformerObject::MaxFallingSpeed()`	Return the maximum falling speed of the object (in pixels per second).
`Object.PlatformerObject::MaxSpeed()`	Return the maximum horizontal speed of the object (in pixels per second).

Destroy when outside of the screen (from extension Destroy Outside Screen Behavior)

Destroy objects automatically when they go outside of the screen's borders. Read more explanations about it.

No expressions for this behavior.

Tiled Sprite (from extension Tiled Sprite Object)

Displays an image repeated over an area. Read more explanations about it.

Expression	Description
`Object.Opacity()`	Opacity

Draggable object (from extension Draggable Behavior)

Move objects by holding a mouse button (or touch). Read more explanations about it.

No expressions for this behavior.

Top-down movement (4 or 8 directions) (from extension Top-down movement)

Move objects left, up, right, and down (and, optionally, diagonally). Read more explanations about it.

Expression	Description
`Object.TopDownMovement::Acceleration()`	Acceleration of the object
`Object.TopDownMovement::Angle()`	Angle, in degrees, of the movement
`Object.TopDownMovement::AngleOffset()`	Rotation offset applied to the object
`Object.TopDownMovement::AngularMaxSpeed()`	Angular maximum speed of the object
`Object.TopDownMovement::Deceleration()`	Deceleration of the object
`Object.TopDownMovement::MaxSpeed()`	Maximum speed of the object
`Object.TopDownMovement::MovementAngleOffset()`	Return the movement angle offset.
`Object.TopDownMovement::Speed()`	Speed of the object
`Object.TopDownMovement::StickAngle()`	Return the angle of the simulated stick input (in degrees)
`Object.TopDownMovement::XVelocity()`	Speed on the X axis of the movement
`Object.TopDownMovement::YVelocity()`	Speed on the Y axis of the movement

Text (from extension Text object)

Displays a text on the screen. Read more explanations about it.

Expression	Description
`Object.Angle()`	Angle
`Object.FontSize()`	Return the font size of a text object.
`Object.Opacity()`	Opacity of a Text object
`Object.Padding()`	Padding
`Object.ScaleX()`	X Scale of a Text object
`Object.ScaleY()`	Y Scale of a Text object
`Object.String()`	Text

Particles emitter (from extension Particle system)

Displays a large number of small particles to create visual effects. Read more

explanations about it.

Expression	Description
`Object.ConeSprayAngle()`	Angle of the spray cone
`Object.CurrentParticleCount()`	Number of particles currently displayed.
`Object.EmitterAngle()`	Emission angle of the particles.
`Object.EmitterForceMax()`	The maximal emission force of the particles.
`Object.EmitterForceMin()`	The minimal emission force of the particles.
`Object.Flow()`	Flow of the particles (particles/second).
`Object.MaxParticlesCount()`	Return the maximum number of displayed particles.
`Object.ParticleAlpha1()`	Start opacity of the particles.
`Object.ParticleAlpha2()`	End opacity of the particles.
`Object.ParticleBlue1()`	The start color blue component of the particles.
`Object.ParticleBlue2()`	The end color blue component of the particles.
`Object.ParticleGravityAngle()`	Angle of gravity.
`Object.ParticleGravityLength()`	Value of gravity.
`Object.ParticleGravityX()`	Gravity of particles applied on X-axis.
`Object.ParticleGravityY()`	Gravity of particles applied on Y-axis.
`Object.ParticleGreen1()`	The start color green component of the particles.
`Object.ParticleGreen2()`	The end color green component of the particles.
`Object.ParticleLifeTimeMax()`	Maximum lifetime of the particles.
`Object.ParticleLifeTimeMin()`	Minimum lifetime of the particles.
`Object.ParticleRed1()`	The start color red component of the particles.
`Object.ParticleRed2()`	The end color red component of the particles.
`Object.ParticleRotationMaxSpeed()`	Return the maximum rotation speed of the particles.
`Object.ParticleRotationMinSpeed()`	Return the minimum rotation speed of the particles.
`Object.ParticleSize1()`	Start size of particles.
`Object.ParticleSize2()`	End size of particles.
`Object.RendererParam1()`	Rendering first parameter
`Object.RendererParam2()`	Rendering second parameter
`Object.Tank()`	Capacity of the particle tank.
`Object.Texture()`	Name of the image displayed by particles.
`Object.ZoneRadius()`	The radius of the emission zone.

Panel Sprite ("9-patch") (from extension Panel Sprite (9-patch) Object)

An image with edges and corners that are stretched separately from the full image. Read more explanations about it.

Expression	Description
`Object.Opacity()`	Opacity

Anchor

Anchor objects to the window's bounds. Read more explanations about it.

No expressions for this behavior.

Shape painter

Allows you to draw simple shapes on the screen Read more explanations about it.

Expression		Description
`Object.FillColorBlue()`		Filing color blue component
`Object.FillColorGreen()`		Filing color green component
`Object.FillColorRed()`		Filing color red component
`Object.FillOpacity()`		Filling opacity
`Object.OutlineColorBlue()`		Outline color blue component
`Object.OutlineColorGreen()`		Outline color green component
`Object.OutlineColorRed()`		Outline color red component
`Object.OutlineOpacity()`		Outline opacity
`Object.OutlineSize()`		Outline size
`Object.ScaleX()`		Return the width's scale of an object.
`Object.ScaleY()`		Return the height's scale of an object.
		X drawing coordinate of a point from the scene
`Object.ToDrawingX(number, number)`	*number*	X scene position
	number	Y scene position
		Y drawing coordinate of a point from the scene
`Object.ToDrawingY(number, number)`	*number*	X scene position
	number	Y scene position
		X scene coordinate of a point from the drawing
`Object.ToSceneX(number, number)`	*number*	X drawing position
	number	Y drawing position
		Y scene coordinate of a point from the drawing
`Object.ToSceneY(number, number)`	*number*	X drawing position
	number	Y drawing position

Text entry (from extension Text entry object)

Invisible object used to get the text entered with the keyboard. Read more explanations about it.

Expression	Description
`Object.String()`	Text entered with keyboard

Inventories

Provides actions and conditions to add an inventory to your game, with items in memory. Read more explanations about it.

Expression	Description	
	Get the number of an item in the inventory	
`Inventory::Count(string, string)`	*string*	Inventory name
	string	Item name

Pathfinding (from extension Pathfinding behavior)

Move objects to a target while avoiding all objects that are flagged as obstacles. Read more explanations about it.

Expression	Description
`Object.Pathfinding::Acceleration()`	Acceleration of the object on the path
`Object.Pathfinding::AngleOffset()`	Rotation offset applied the object on the path
`Object.Pathfinding::AngularMaxSpeed()`	Angular maximum speed of the object on the path
`Object.Pathfinding::CellHeight()`	Height of the virtual grid
`Object.Pathfinding::CellWidth()`	Width of the virtual grid
`Object.Pathfinding::DestinationX()`	Destination X position
`Object.Pathfinding::DestinationY()`	Destination Y position
`Object.Pathfinding::ExtraBorder()`	Extra border applied the object on the path
`Object.Pathfinding::GetNodeX(number)`	Get next waypoint X position *number* Node index (start at 0!)
`Object.Pathfinding::GetNodeY(number)`	Get next waypoint Y position *number* Node index (start at 0!)
`Object.Pathfinding::GridOffsetX()`	Return X offset of the virtual grid.
`Object.Pathfinding::GridOffsetY()`	Return Y offset of the virtual grid.
`Object.Pathfinding::LastNodeX()`	Last waypoint X position
`Object.Pathfinding::LastNodeY()`	Last waypoint Y position
`Object.Pathfinding::MaxSpeed()`	Maximum speed of the object on the path
`Object.Pathfinding::MovementAngle()`	Angle of movement on its path
`Object.Pathfinding::NextNodeIndex()`	Get the index of the next waypoint to reach
`Object.Pathfinding::NextNodeX()`	Get next waypoint X position
`Object.Pathfinding::NextNodeY()`	Get next waypoint Y position
`Object.Pathfinding::NodeCount()`	Get the number of waypoints on the path
`Object.Pathfinding::Speed()`	Speed of the object on the path

Obstacle for pathfinding (from extension Pathfinding behavior)

Flag objects as being obstacles for pathfinding. Read more explanations about it.

Expression	Description
`Object.PathfindingObstacle::Cost()`	Obstacle cost

Physics Engine (from extension Physics Engine (deprecated))

Make objects move as if they are subject to the laws of physics. If you're creating a new game, prefer Physics Engine 2.0 Read more explanations about it.

Expression	Description
`Object.Physics::AngularDamping()`	Angular damping
`Object.Physics::AngularVelocity()`	Angular speed
`Object.Physics::LinearDamping()`	Linear damping
`Object.Physics::LinearVelocity()`	Linear speed
`Object.Physics::LinearVelocityX()`	X component
`Object.Physics::LinearVelocityY()`	Y component
`Object.Physics::PolygonScaleX()`	Collision polygon X scale
`Object.Physics::PolygonScaleY()`	Collision polygon Y scale

Advanced window management

Provides advanced features related to the game window positioning and interaction with the operating system.

Expression	Description
`AdvancedWindow::WindowOpacity()`	Returns the current window opacity (a number from 0 to 1, 1 being fully opaque).
`AdvancedWindow::WindowX()`	Returns the current window X position.
`AdvancedWindow::WindowY()`	Returns the current window Y position.

BBText (from extension BBCode Text Object)

Displays a rich text label using BBCode markup (allowing to set parts of the text as bold, italic, use different colors and shadows). Read more explanations about it.

Expression	Description
`Object.GetBBText()`	Get BBCode text
`Object.GetFontFamily()`	Get the base font family
`Object.GetFontSize()`	Get the base font size
`Object.GetOpacity()`	Get the base opacity
`Object.GetWrappingWidth()`	Get the wrapping width

Bitmap Text

Displays a text using a “Bitmap Font” (an image representing characters). This is more performant than a traditional Text object and it allows for complete control on the characters aesthetic. Read more explanations about it.

Expression	Description
`Object.Alignment()`	Return the text alignment.
`Object.FontName()`	Return the font name (defined in the Bitmap font).
`Object.FontSize()`	Return the font size, defined in the Bitmap Font.
`Object.Opacity()`	Return the opacity, between 0 (fully transparent) and 255 (opaque).
`Object.Scale()`	Return the scale (1 by default).
`Object.Text()`	Return the text.
`Object.WrappingWidth()`	Return the width, in pixels, after which the text is wrapped on next line.

Device sensors

Allow the game to access the sensors of a mobile device. Read more explanations about it.

Expression	Description
`DeviceSensors::AccelerationX()`	Get the devices acceleration on the X-axis (m/s²)
`DeviceSensors::AccelerationY()`	Get the devices acceleration on the Y-axis (m/s²)
`DeviceSensors::AccelerationZ()`	Get the devices acceleration on the Z-axis (m/s²)
`DeviceSensors::OrientationAbsolute()`	Get if the devices orientation is absolute and not relative
`DeviceSensors::OrientationAlpha()`	Get the devices orientation Alpha (compass)
`DeviceSensors::OrientationBeta()`	Get the devices orientation Beta
`DeviceSensors::OrientationGamma()`	Get the devices orientation Gamma value
`DeviceSensors::RotationAlpha()`	Get the devices rotation Alpha
`DeviceSensors::RotationBeta()`	Get the devices rotation Beta
`DeviceSensors::RotationGamma()`	Get the devices rotation Gamma

Dialogue Tree (experimental)

Handle dialogue trees, made using Yarn Spinner. Useful to make complex dialogues with multiple choices. The Yarn Spinner editor is embedded in GDevelop so you can edit your dialogues without leaving GDevelop. Read more explanations about it.

Expression	Description
`DialogueTree::BranchTag(number)`	Get a tag of the current branch of the running dialogue via its index *number* Tag Index Number
`DialogueTree::BranchTags()`	Get the tags of the current branch of the running dialogue
`DialogueTree::BranchText()`	Get the full raw text of the current branch

Expression	Description
`DialogueTree::BranchTitle()`	Get the title of the current branch of the running dialogue
`DialogueTree::ClippedLineText()`	Get dialogue line text clipped by the typewriter effect. Use the “Scroll clipped text” action to control the typewriter effect.
`DialogueTree::CommandParameter(number)`	Get the parameters of a command call - «command withParameter anotherParameter» *number* parameter Index Number *Optional*.
`DialogueTree::CommandParametersCount()`	Get the number of parameters in the currently passed command
`DialogueTree::HorizontalOptionsList(string)`	Get the text of all available options from an Options line type as a horizontal list. You can also pass the selected option's cursor string, which by default is → *string* Options Selection Cursor
`DialogueTree::LineText()`	Returns the current dialogue line text
`DialogueTree::Option(number)`	Get the text of an option from an Options line type, using the option's Number. The numbers start from 0. *number* Option Index Number
`DialogueTree::OptionsCount()`	Get the number of options in an options line type
`DialogueTree::SelectedOptionIndex()`	Get the number of the currently selected option. Use this to help you render the option selection marker at the right place.
`DialogueTree::TagParameter(number)`	Get parameter from a Tag found by the branch contains tag condition *number* parameter Index Number *Optional*.
`DialogueTree::Variable(string)`	Get dialogue state value *string* Variable Name
`DialogueTree::VerticalOptionsList(string)`	Get the text of all available options from an Options line type as a vertical list. You can also pass the selected option's cursor string, which by default is → *string* Options Selection Cursor
`DialogueTree::VisitedBranchTitles()`	Get a list of all visited branches

Facebook Instant Games

Allow your game to send scores and interact with the Facebook Instant Games platform. Read more explanations about it.

Expression	Description

Expression	Description
`FacebookInstantGames::PlayerId()`	Get the player unique identifier
`FacebookInstantGames::PlayerName()`	Get the player name

File system

Access the filesystem of the operating system. Read more explanations about it.

Expression	Description
`FileSystem::DesktopPath()`	Get the path to the desktop folder.
`FileSystem::DirectoryName(string)`	Returns the portion of the path that represents the directories, without the ending file name. *string* File or folder path
`FileSystem::DocumentsPath()`	Get the path to the documents folder.
`FileSystem::ExecutableFolderPath()`	Get the path to this game executable folder.
`FileSystem::ExecutablePath()`	Get the path to this game executable file.
`FileSystem::ExtensionName(string)`	Returns the extension of the file designated by the given path, including the extension period. For example: ”.txt“. *string* File path
`FileSystem::FileName(string)`	Returns the name of the file with its extension, if any. *string* File path
`FileSystem::PathDelimiter()`	Get the operating system path delimiter.
`FileSystem::PicturesPath()`	Get the path to the pictures folder.
`FileSystem::TempPath()`	Get the path to temp folder.
`FileSystem::UserHomePath()`	Get the path to the user home folder.
`FileSystem::UserdataPath()`	Get the path to userdata folder (for application settings).

Firebase

Use Google Firebase services (database, functions, storage...) in your game. Read more explanations about it.

Expression	Description
`Firebase::GetAccountCreationTime()`	Gets the accounts creation time.
`Firebase::GetAuthToken(string)`	Get the user authentifiifcation token. The token is the proof of authentication. *string* Setting Name
`Firebase::GetLastLoginTime()`	Gets the user last login time.
`Firebase::GetPhoneNumber()`	Gets the user phone number.
`Firebase::GetPhotoURL()`	Gets an URL to the user profile picture.
`Firebase::GetRefreshToken()`	Gets the user refresh token. For advanced usage only.

Expression	Description
`Firebase::GetRemoteConfigNumber(string)`	Get a setting from Firebase Remote Config as Number. *string* Setting Name
`Firebase::GetRemoteConfigString(string)`	Get a setting from Firebase Remote Config as a string. *string* Setting Name
`Firebase::GetTenantID()`	Gets the user tenant ID. For advanced usage only.
`Firebase::GetUserDisplayName()`	Gets the user display name.
`Firebase::GetUserEmail()`	Gets the user email address.
`Firebase::GetUserUID()`	Gets the user Unique IDentifier. Use that to link data to an user instead of the name or email.
`Firebase::ServerTimestamp()`	Set a field to the timstamp on the server when the request arrives there

---

Leaderboards (experimental)

Allow your game to send scores to your leaderboards. Read more explanations about it.

Expression	Description
`Leaderboards::FormatPlayerName(string)`	Formats a name so that it can be submitted to a leaderboard. *string* Raw player name
`Leaderboards::LastSaveError(leaderboardId)`	Get the error of the last save attempt. *leaderboardId* Leaderboard If no leaderboard is specified, will return the value related to the last leaderboard save action. *Optional*.

Light (from extension Lights)

Displays a light on the scene, with a customizable radius and color. Add then the Light Obstacle behavior to the objects that must act as obstacle to the lights.

No expressions for this object.

Light Obstacle Behavior (from extension Lights)

Flag objects as being obstacles to light. The light emitted by light objects will be stopped by the object.

No expressions for this behavior.

P2P (experimental)

Allow game instances to communicate remotely using messages sent via WebRTC (P2P). Read more explanations about it.

Expression	Description
`P2P::GetEventData(string)`	Returns the data received when the specified event was last triggered *string* Event name
`P2P::GetEventSender(string)`	Returns the id of the peer that triggered the event *string* Event name
`P2P::GetID()`	Gets the client ID of the current game instance
`P2P::GetLastConnectedPeer()`	Gets the ID of the newly connected peer.
`P2P::GetLastDisconnectedPeer()`	Gets the ID of the latest peer that has disconnected.
`P2P::GetLastError()`	Gets the description of the last P2P error

Physics Engine 2.0

Simulate realistic object physics with gravity, forces, joints, etc. Read more explanations about it.

Expression	Description
`Object.Physics2::AngularDamping()`	Get the angular damping of an object.
`Object.Physics2::AngularVelocity()`	Get the angular velocity of an object.
`Object.Physics2::Density()`	Get the density of an object.
`Object.Physics2::DistanceJointDampingRatio(number)`	Distance joint damping ratio *number* Joint ID
`Object.Physics2::DistanceJointFrequency(number)`	Distance joint frequency *number* Joint ID
`Object.Physics2::DistanceJointLength(number)`	Distance joint length *number* Joint ID
`Object.Physics2::Friction()`	Get the friction of an object.
`Object.Physics2::FrictionJointMaxForce(number)`	Friction joint maximum force *number* Joint ID
`Object.Physics2::FrictionJointMaxTorque(number)`	Friction joint maximum torque *number* Joint ID
`Object.Physics2::GearJointFirstJoint(number)`	Gear joint first joint *number* Joint ID
`Object.Physics2::GearJointRatio(number)`	Gear joint ratio *number* Joint ID
`Object.Physics2::GearJointSecondJoint(number)`	Gear joint second joint *number* Joint ID
`Object.Physics2::GravityScale()`	Get the gravity scale of an object.

Expression	Description
`Object.Physics2::GravityX()`	World gravity on X axis
`Object.Physics2::GravityY()`	World gravity on Y axis
`Object.Physics2::Inertia()`	Return the rotational inertia of the object (in kilograms * meters * meters)
`Object.Physics2::JointFirstAnchorX(number)`	Joint first anchor X *number* Joint ID
`Object.Physics2::JointFirstAnchorY(number)`	Joint first anchor Y *number* Joint ID
`Object.Physics2::JointReactionForce(number)`	Joint reaction force *number* Joint ID
`Object.Physics2::JointReactionTorque(number)`	Joint reaction torque *number* Joint ID
`Object.Physics2::JointSecondAnchorX(number)`	Joint second anchor X *number* Joint ID
`Object.Physics2::JointSecondAnchorY(number)`	Joint second anchor Y *number* Joint ID
`Object.Physics2::LinearDamping()`	Get the linear damping of an object.
`Object.Physics2::LinearVelocity()`	Get the linear velocity of an object.
`Object.Physics2::LinearVelocityX()`	Get the linear velocity of an object on X axis.
`Object.Physics2::LinearVelocityY()`	Get the linear velocity of an object on Y axis.
`Object.Physics2::Mass()`	Return the mass of the object (in kilograms)
`Object.Physics2::MassCenterX()`	Mass center X
`Object.Physics2::MassCenterY()`	Mass center Y
`Object.Physics2::MotorJointAngularOffset(number)`	Motor joint angular offset *number* Joint ID
`Object.Physics2::MotorJointCorrectionFactor(number)`	Motor joint correction factor *number* Joint ID
`Object.Physics2::MotorJointMaxForce(number)`	Motor joint maximum force *number* Joint ID
`Object.Physics2::MotorJointMaxTorque(number)`	Motor joint maximum torque *number* Joint ID
`Object.Physics2::MotorJointOffsetX(number)`	Motor joint offset X *number* Joint ID
`Object.Physics2::MotorJointOffsetY(number)`	Motor joint offset Y *number* Joint ID
`Object.Physics2::MouseJointDampingRatio(number)`	Mouse joint damping ratio *number* Joint ID
`Object.Physics2::MouseJointFrequency(number)`	Mouse joint frequency

Expression	Description
	number Joint ID
`Object.Physics2::MouseJointMaxForce(number)`	Mouse joint maximum force *number* Joint ID
`Object.Physics2::MouseJointTargetX(number)`	Mouse joint target X *number* Joint ID
`Object.Physics2::MouseJointTargetY(number)`	Mouse joint target Y *number* Joint ID
`Object.Physics2::PrismaticJointAxisAngle(number)`	Prismatic joint axis angle *number* Joint ID
`Object.Physics2::PrismaticJointMaxMotorForce(number)`	Prismatic joint maximum motor force *number* Joint ID
`Object.Physics2::PrismaticJointMaxTranslation(number)`	Prismatic joint maximum translation *number* Joint ID
`Object.Physics2::PrismaticJointMinTranslation(number)`	Prismatic joint minimum translation *number* Joint ID
`Object.Physics2::PrismaticJointMotorForce(number)`	Prismatic joint motor force *number* Joint ID
`Object.Physics2::PrismaticJointMotorSpeed(number)`	Prismatic joint motor speed *number* Joint ID
`Object.Physics2::PrismaticJointReferenceAngle(number)`	Prismatic joint reference angle *number* Joint ID
`Object.Physics2::PrismaticJointSpeed(number)`	Prismatic joint speed *number* Joint ID
`Object.Physics2::PrismaticJointTranslation(number)`	Prismatic joint current translation *number* Joint ID
`Object.Physics2::PulleyJointFirstGroundAnchorX(number)`	Pulley joint first ground anchor X *number* Joint ID
`Object.Physics2::PulleyJointFirstGroundAnchorY(number)`	Pulley joint first ground anchor Y *number* Joint ID
`Object.Physics2::PulleyJointFirstLength(number)`	Pulley joint first length *number* Joint ID
`Object.Physics2::PulleyJointRatio(number)`	Pulley joint ratio *number* Joint ID
`Object.Physics2::PulleyJointSecondGroundAnchorX(number)`	Pulley joint second ground anchor X *number* Joint ID
`Object.Physics2::PulleyJointSecondGroundAnchorY(number)`	Pulley joint second ground anchor Y *number* Joint ID
`Object.Physics2::PulleyJointSecondLength(number)`	Pulley joint second length

Expression	Description
	number Joint ID
`Object.Physics2::Restitution()`	Get the restitution of an object.
`Object.Physics2::RevoluteJointAngle(number)`	Revolute joint current angle *number* Joint ID
`Object.Physics2::RevoluteJointMaxAngle(number)`	Revolute joint maximum angle *number* Joint ID
`Object.Physics2::RevoluteJointMaxMotorTorque(number)`	Revolute joint maximum motor torque *number* Joint ID
`Object.Physics2::RevoluteJointMinAngle(number)`	Revolute joint minimum angle *number* Joint ID
`Object.Physics2::RevoluteJointMotorSpeed(number)`	Revolute joint motor speed *number* Joint ID
`Object.Physics2::RevoluteJointMotorTorque(number)`	Revolute joint motor torque *number* Joint ID
`Object.Physics2::RevoluteJointReferenceAngle(number)`	Revolute joint reference angle *number* Joint ID
`Object.Physics2::RevoluteJointSpeed(number)`	Revolute joint angular speed *number* Joint ID
`Object.Physics2::RopeJointMaxLength(number)`	Rope joint maximum length *number* Joint ID
`Object.Physics2::TimeScale()`	World time scale
`Object.Physics2::WeldJointDampingRatio(number)`	Weld joint damping ratio *number* Joint ID
`Object.Physics2::WeldJointFrequency(number)`	Weld joint frequency *number* Joint ID
`Object.Physics2::WeldJointReferenceAngle(number)`	Weld joint reference angle *number* Joint ID
`Object.Physics2::WheelJointAxisAngle(number)`	Wheel joint axis angle *number* Joint ID
`Object.Physics2::WheelJointDampingRatio(number)`	Wheel joint damping ratio *number* Joint ID
`Object.Physics2::WheelJointFrequency(number)`	Wheel joint frequency *number* Joint ID
`Object.Physics2::WheelJointMaxMotorTorque(number)`	Wheel joint maximum motor torque *number* Joint ID
`Object.Physics2::WheelJointMotorSpeed(number)`	Wheel joint motor speed *number* Joint ID
`Object.Physics2::WheelJointMotorTorque(number)`	Wheel joint motor torque *number* Joint ID
`Object.Physics2::WheelJointSpeed(number)`	Wheel joint speed *number* Joint ID

Expression	Description
`Object.Physics2::WheelJointTranslation(number)`	Wheel joint current translation *number* Joint ID

Player Authentication (experimental)

Allow your game to authenticate players. Read more explanations about it.

Expression	Description
`PlayerAuthentication::Username()`	Get the username of the authenticated player.

Text input (experimental) (from extension Text Input)

A text field the player can type text into.

Expression	Description
`Object.BorderOpacity()`	Return the border opacity, between 0 (fully transparent) and 255 (opaque).
`Object.BorderWidth()`	Return the border width.
`Object.FillOpacity()`	Return the fill opacity, between 0 (fully transparent) and 255 (opaque).
`Object.Font size()`	Return the font size.
`Object.FontResourceName()`	Return the font name.
`Object.InputType()`	Return the input type.
`Object.Opacity()`	Return the opacity, between 0 (fully transparent) and 255 (opaque).
`Object.Placeholder()`	Return the placeholder.
`Object.Text()`	Return the text.

Tilemap collision mask (experimental) (from extension Tilemap)

Invisible object handling collisions with parts of a tilemap. Read more explanations about it.

Expression	Description
`Object.ScaleX()`	Return the width's scale of an object.
`Object.ScaleY()`	Return the height's scale of an object.

Tilemap

Displays a tiled-based map, made with the Tiled editor (download it separately on https://www.mapeditor.org/). Read more explanations about it.

Expression	Description
`Object.AnimationFps()`	Get the animation speed (in frames per second)
`Object.AnimationSpeedScale()`	Get the Animation speed scale
`Object.LayerIndex()`	Get the layer index being displayed
`Object.ScaleX()`	Return the width's scale of an object.
`Object.ScaleY()`	Return the height's scale of an object.

Tweening

Animate object properties over time. This allows smooth transitions, animations or movement of objects to specified positions. Read more explanations about it.

Expression	Description	
`Tween::Ease(string, number, number, number)`	Tween between 2 values according to an easing function.	
	string	Easing
	number	From value
	number	To value
	number	Weighting From 0 to 1.

Tween (from extension Tweening)

Smoothly animate position, angle, scale and other properties of objects. Read more explanations about it.

Expression	Description	
`Object.Tween::Progress(identifier)`	Progress of a tween (between 0.0 and 1.0)	
	identifier	Tween Identifier

Video

Displays a video. Read more explanations about it.

Expression	Description
`Object.CurrentTime()`	Return the current time of a video object (in seconds).
`Object.Duration()`	Return the duration of a video object (in seconds).
`Object.Opacity()`	Return the opacity of a video object
`Object.PlaybackSpeed()`	Return the playback speed of a video object
`Object.Volume()`	Get the volume of a video object, between 0 (muted) and 100 (maximum).