

Scott Tanner | Chris Stapleton

Official support Discord
https://discord.gg/mVgdh49xxP



Introduction

The in-game cinematic is a powerful storytelling device in games, able to reveal new information to the player and communicate story. By doing this within the gameworld, instead of showing a pre-rendered video, you can maintain the player's immersion and save both disk space and video memory.

Cine|Guru MAX is a revolutionary toolkit of entities and scripts for GameGuru MAX that empowers users to create their own in-game cinematics. It offers a huge degree of control over cameras, actors, audio, and graphics. Building on the success of the original **Cine|Guru** for GameGuru Classic, we've improved our toolkit to harness the latest usability features of GameGuru MAX, making it even easier to use.

This guide will show you how to get the most out of **Cine|Guru MAX** so you too can create captivating cinematics for your GameGuru MAX games. Each step is supplemented by a video tutorial which you can follow along with, and we've included some example fpms so you can study many of the setups detailed in this guide.

Contents

Cameras

5 | Static cameras

7 | Simple moving cameras

9 | Complex moving cameras

11 | Multi-camera example

Focus

13 | Focal points

14 | Nodal focal points

17 | Focal targets

Lights

19 | Light markers

20 | Light nodes

21 | Using focal points with light markers

22 | Activating light markers

Actors

24 | Actors

25 | Marks

26 | Action text

28 | Looking up animations

29 | Writing action text

30 | Moving and talking

31 | Music commands

32 | Subtitles

34 | Activating and triggering

35 | Switching and aborting

Triggers

37 | Cine-trigger zones

38 | Cine-trigger entities

39 | Generic triggers

40 | Camera shake triggers

42 | Image triggers

44 | Sound triggers

45 | Credits triggers

Preparing for standalone

48 | Fpp files

49 | Cinematics as videos


CAMERAS

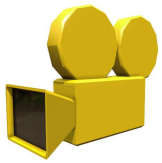
Cameras
- **Static cameras**
- Simple moving cameras
- Complex moving cameras
- Multi-camera example

Focus
Lights
Actors
Triggers
Preparing for standalone

Static cameras

▶ **Video Tutorial:** https://youtu.be/RJ4TARLMY-c

In the original **Cine|Guru**, we used the names of each entity to give it specific instructions and define parameters.
In **Cine|Guru MAX**, we utilise GameGuru MAX's dynamic lua and visual logic systems.



Camera Marker

To create a camera, simply place a camera marker entity in your level. When used by themselves, camera markers will produce static (non-moving) shots. Each camera marker has parameters you can tweak in the *Behavior* pane of the GameGuru editor. These are:



Filmtime | The duration of the shot in seconds, e.g. 5.000 = 5 seconds.

Fade | The duration of a fade in from black at the start of the shot. When used with static cameras, it will only add a fade in from black at the start of the shot.

Fls | Focal length start – The field of view at the start of the shot.

Fle | Focal length end – The field of view at the end of the shot.

Let's create a simple cinematic consisting of two shots from static cameras.



We will trigger the cinematic using a cine-trigger zone. We'll cover these in more detail in the Triggers chapter.

Our first camera marker (1) has a **filmtime** of 10, meaning the shot will last 10-seconds, starting with a **fade** in from black of 2-seconds. The shot begins with an **fls** of 120, but ends with an **fle** of 100, meaning it will gradually zoom in over the course of the shot. At the end of the shot, we cut to the next camera marker.

Camera marker (2) has a 4-second **filmtime**, with a **fade** of 0, so there is no fade at the beginning of this shot. When using static cameras, the fade can only be at the beginning of a shot and not at the end. The **fls** and **fle** are both 90, so the focal length doesn't change. Once this shot has finished, the cinematic is over.

Cameras
- **Static cameras**
- Simple moving cameras
- Complex moving cameras
- Multi-camera example

Focus

Lights

Actors

Triggers

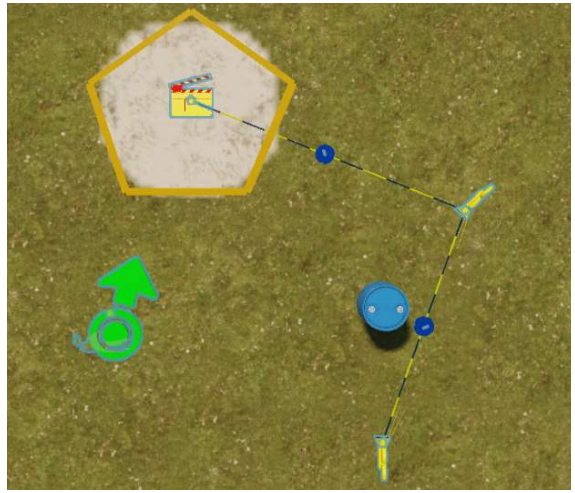
Preparing for standalone

Before our cinematic will work, we need to connect all the elements together. To do this, we will use GameGuru MAX's visual logic system. To access the visual logic editing mode, you click the *Visual Logic Connections* icon in the editor.

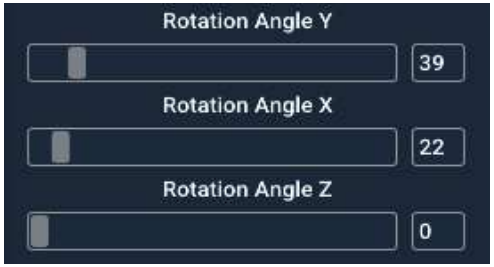


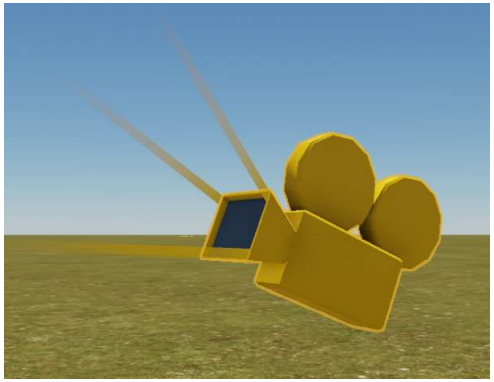
Using the visual logic system

Simply draw a connection between the cine-trigger zone and the first camera marker, and then from the first camera marker to the second camera marker. Your connections should look like the example on the right. If you make a mistake, click on the blue blob in the middle of the connection and then click the *Delete Relationship* button.



Click *Test Game*, enter your trigger zone and watch your cinematic play out.





Rotation

How do you tell your cameras what to point at?

The simplest method is to use GameGuru' MAX's rotation sliders, which can be found in each entity's *Positioning, Rotating and Scaling* pane of the editor, or by hitting F3 to bring up the rotation gizmo.

The Angle Y slider adjusts the camera's yaw (or “pan” in filmmaking speak), whilst the Angle X slide adjusts its pitch (“tilt”). Angle Z can be used to roll the camera, but needs to be used carefully in combination with Angle X.

In the Focus chapter, we'll look at other ways to manipulate what the camera points at, including tracking moving objects.

Cameras
- Static cameras
- Simple moving cameras
- Complex moving cameras
- Multi-camera example

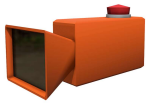
Focus
Lights
Actors
Triggers
Preparing for standalone

Simple moving cameras

▶ **Video Tutorial:** https://youtu.be/5PmsORoY6jA

Whilst static cameras can be very effective, moving cameras can be used to track action, reveal information to the player, and dramatically add production value. With **Cine|Guru MAX**, moving the camera is as simple as placing a endpoint marker and linking it to your camera.

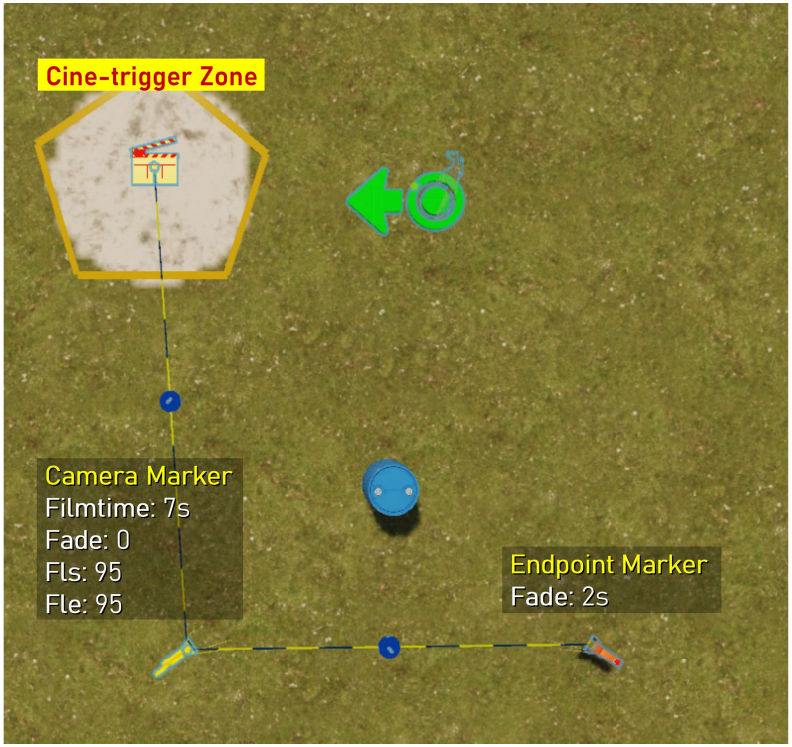




Endpoint Marker
To create a simple camera move, place a endpoint marker in your level. Then, making sure you're in visual logic mode, draw a connection between your camera marker and your endpoint marker. Endpoint markers only have one tweakable parameter.

Fade | The duration of a fade to black at the end of the shot, in seconds.

Let's create a simple camera move using a camera marker and an endpoint.



We're triggering the cinematic using a cine-trigger zone. We'll look at triggers in a later chapter. The zone is connected to our camera marker.

We've set up our camera marker to make the shot last 7 seconds. We set **Fade** to 0, so the shot won't start with a fade in from black, and the focal length will be 95 throughout.

By connecting our camera marker to the endpoint marker, we are telling GameGuru MAX to move the camera to this endpoint when the shot is activated. As the shot is 7 seconds long, it will take 7 seconds to move from the position and orientation of the camera marker to the position and orientation of the endpoint.

At the endpoint, we've specified a 2 second **fade** to black during the last 2 seconds of the shot. In this example, we haven't specified a next camera, so once the shot ends, so does the cinematic and control is returned to the player.

Cameras
- Static cameras
- **Simple moving cameras**
- Complex moving cameras
- Multi-camera example

Focus

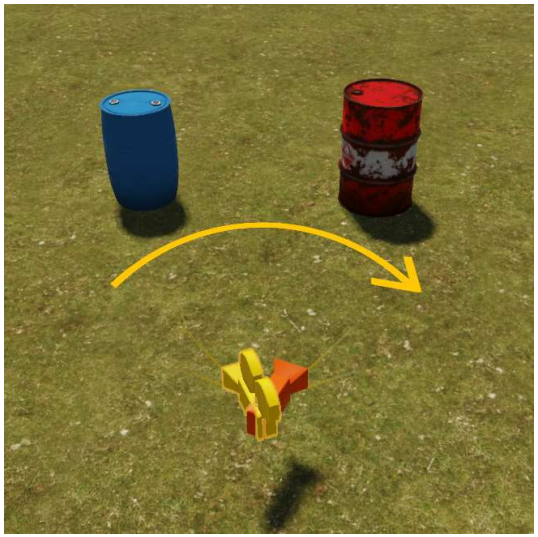
Lights

Actors

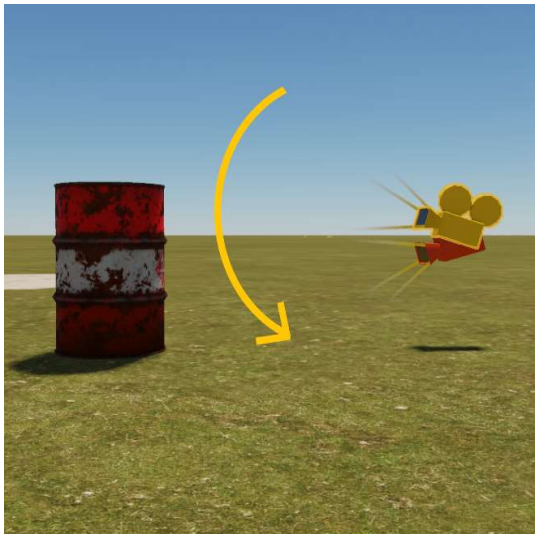
Triggers

Preparing for standalone

Once you've mastered the use of camera endpoints, you'll be able to recreate lots of popular camera moves with ease. Why not try creating some of these yourself to get more practice in creating cinematic motion.



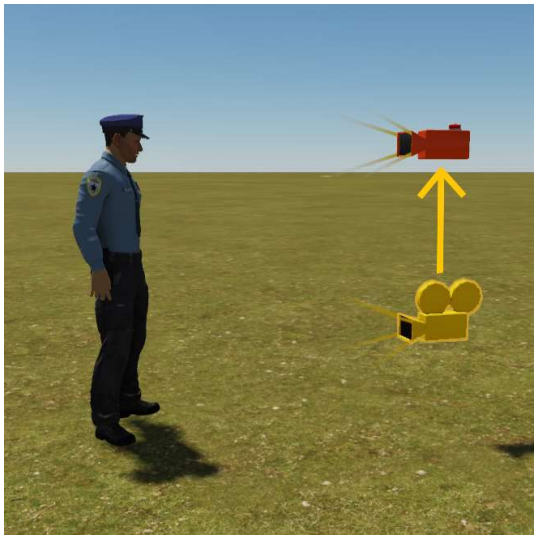
Pan between subjects



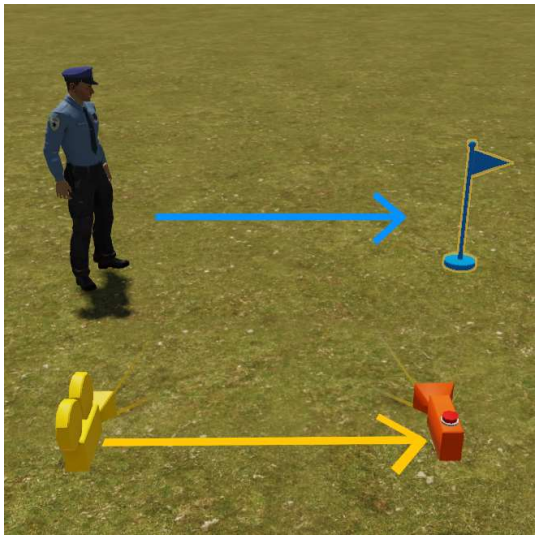
Tilt up or down between subjects



Push in on a subject (or **Pull out**)



Jib the camera up or down



Dolly alongside a moving subject

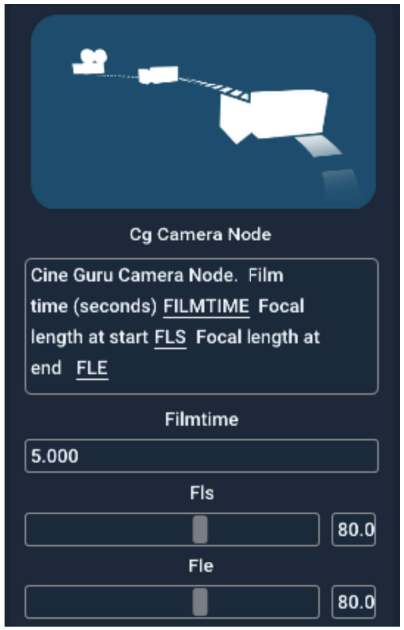
Complex moving cameras

▶ **Video Tutorial:** https://youtu.be/D-uhA4z0e00

Using endpoints will create simple A-to-B camera moves but for more complex motions, such as along paths, you can use camera nodes between your camera and endpoint and have your camera move between them.



Camera Node
By connecting one or more camera nodes between your camera marker and endpoint marker, you can create a path for your camera to follow.
Camera nodes have the following editable parameters:

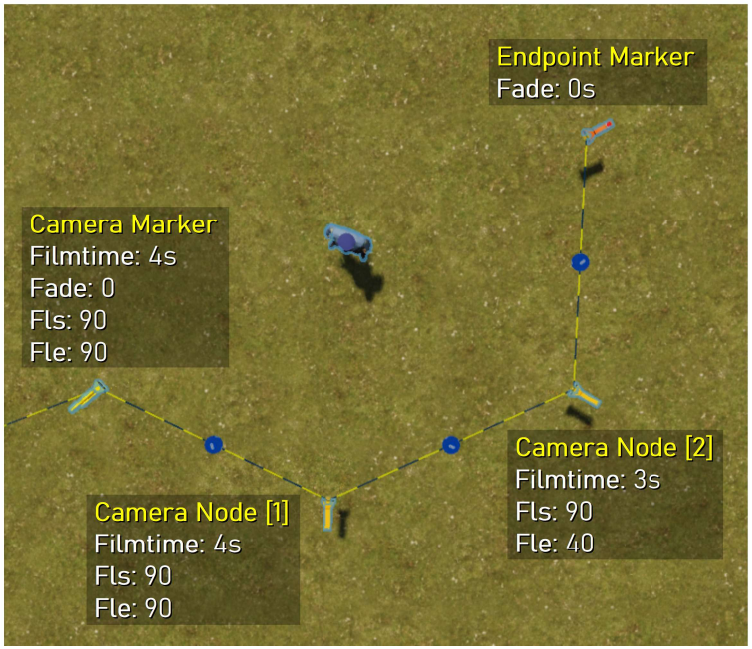


Filmtime | The duration of the next phase of the shot, e.g. 5.000 = 5 seconds.

Fls | Focal length start – The field of view at the start of this phase.

Fle | Focal length end – The field of view at the end of this phase.

It's important to remember that using camera nodes changes how the **Filmtime** parameter works for the camera marker; instead of setting the duration of the whole shot, now it only sets the time taken to travel to the first node.



In this example, our camera will take 4 seconds to travel to node 1. It will automatically move and rotate from the position and orientation of the camera marker to the position and orientation of node 1.

Unlike a simple A-to-B camera move, on paths with multiple nodes, the camera script calculates a smooth curve to follow between nodes, resulting in smoother motion instead of a series of straight lines.

Once at node 1, the camera will instantly continue on to node 2, again taking 4 seconds to get there. At node 2, the camera continues to the next node, which in this case is our endpoint. It will take 3 seconds to get there, notice also there is zoom in from 90 to 40; by the time the camera reaches the endpoint, the FOV will be 40.

Note: the camera does not slow to a stop at the endpoint.

Cameras
- Static cameras
- Simple moving cameras
- **Complex moving cameras**
- Multi-camera example

Focus

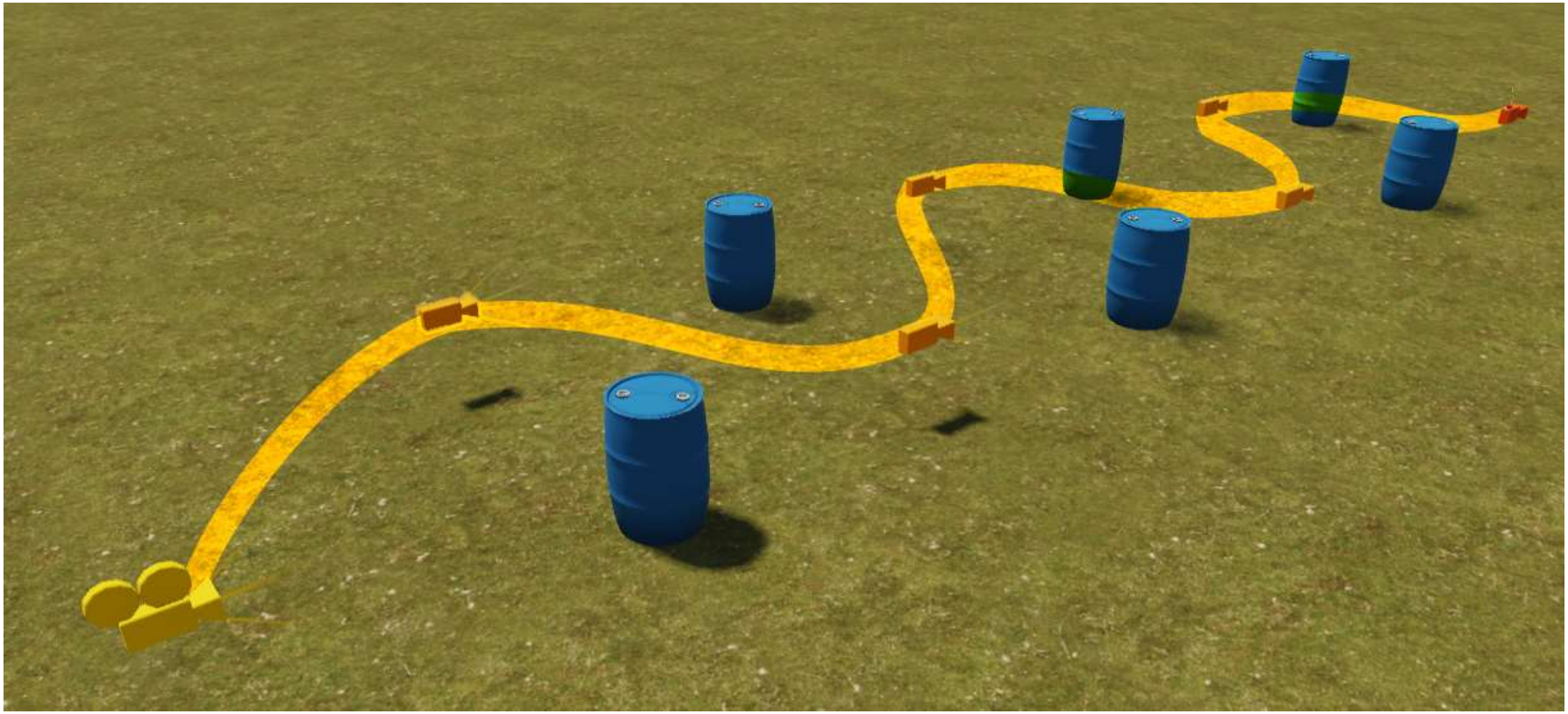
Lights

Actors

Triggers

Preparing for standalone

Nodal paths can be used to create all sorts of arcing and sweeping camera moves – your only limit is your creativity!



The camera script calculates its path to the next node as a smooth curve in three dimensions, meaning you can create fluid movements that rise or fall in height too...perhaps both in the same shot!

As with simple A-to-B camera moves, the script will gradually rotate the camera in all three axes to match the orientation of the next node. Breathtaking fly-throughs of your level are now possible thanks to **Cine**|Guru **MAX**.

Multi-camera example

Now that you've been introduced to all three types of camera marker in **Cine**|Guru **MAX**, let's build a simple scene to show how they can be used together. To keep things straightforward, our subject will be a static object; in this case, a box.



Trigger & camera 1

Our cine-trigger zone is connected to our first camera. As the player enters the zone, the camera activates; there's no camera nodes associated with this camera, which means it's a static shot. It lasts 4 seconds, with no fade in at the beginning, and has a zoom in from FOV 100 to 70. At the end of the shot, we've connected it to the next camera to cut to.

Camera 2

This is a simple moving camera, a slow push in towards the box. The shot lasts 4 seconds, with no fade, and has a fixed FOV of 80 throughout. Our camera marker here will travel to the endpoint it's connected to. The endpoint does not specify a fade to black. The endpoint is connected to our third and final camera marker.

Camera 3

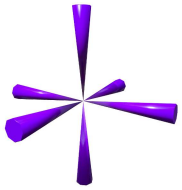
Our third camera travels along a nodal path around the box. It will take 2.5 seconds to travel from its initial position to the first node. The camera node instructs what the camera will do next; travel to its endpoint in 2.5 seconds. The endpoint specifies a 2 second fade to black during the last 2 seconds of the shot. Once the camera reaches the endpoint, the cinematic will end and control will be returned to the player.


FOCUS

Focal points

▶ **Video Tutorial:** https://youtu.be/4X_ti2lY8Oc

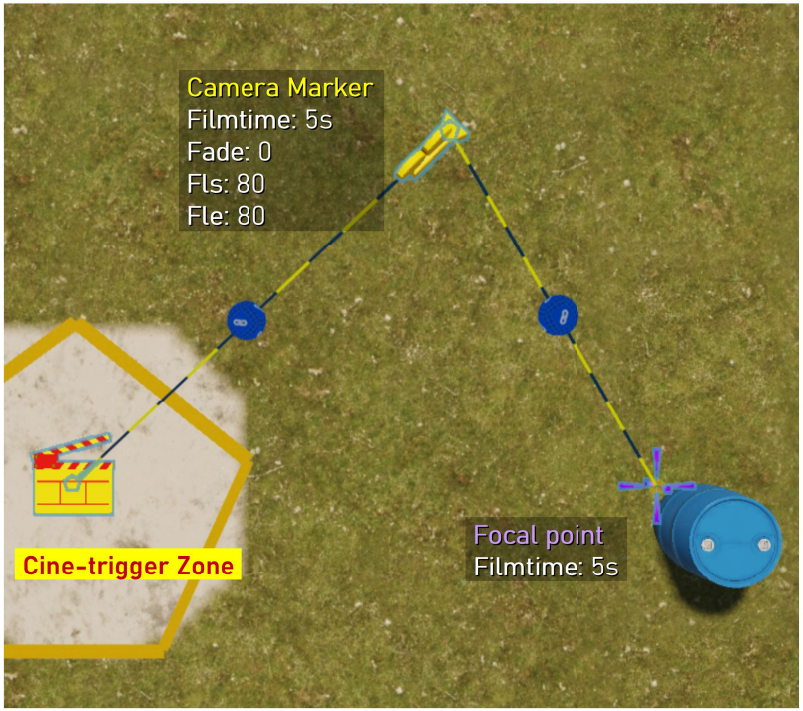
In the previous chapter, we introduced camera markers, endpoints, and nodes, using GameGuru MAX's rotation sliders to angle them at our subjects. However, this method is crude and imprecise, and makes tracking moving subjects difficult. Instead, **Cine|Guru MAX** features three helpful alternatives to help you frame your shots, and even track moving subjects. In this chapter, we'll look at each method available, starting with focal points.



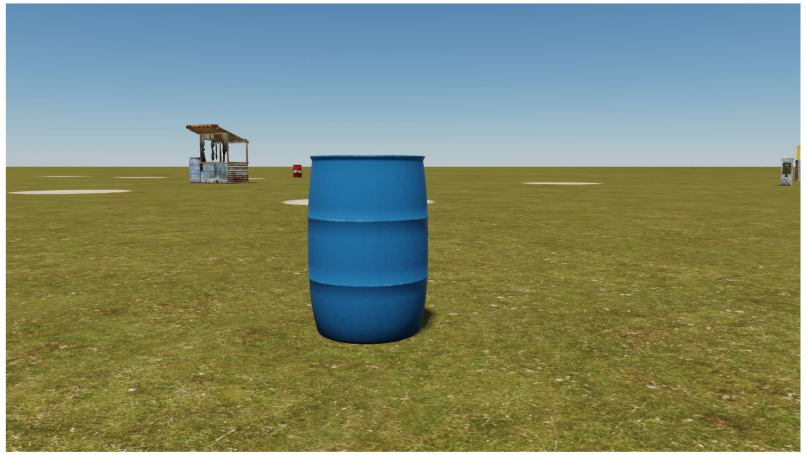
Focal Point
Connecting your camera, node, or endpoint to a focal point will point the camera towards the centre of the focal point, where the cones converge. You can think of a focal point as a three-dimensional crosshair. It has one editable parameter, which we'll explain in more detail later:

Filmtime | The transition time to the next focal point.

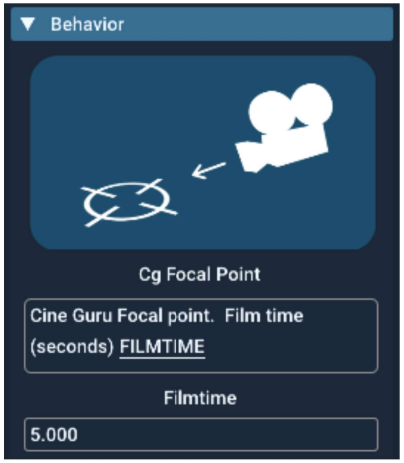
Focal points are invisible in-game, unaffected by physics, and work at any distance from the camera. You can place them anywhere in the level, including in mid-air, and they will hang there in-game.



In this example, we've placed a camera marker down pointing in totally the wrong direction. But we've connected it to a focal point in front of our barrel. The focal point overrides any rotations we've given the camera marker in the editor and makes the camera point directly towards it. Here's the view our camera will show in-game, when activated:



Nodal focal points



Focal points can also be chained together like nodes, creating a “path of focus”. When activated, the camera's focus will smoothly travel along this path as instructed, resulting in the camera turning. This can be used as an alternative way to create simple camera moves like pans or tilts, but also in more creative ways, such as mimicking a person's point of view.

The one editable parameter, **Filmtime**, accessed via the *Behavior* pane, specifies how long it takes to transition focus from the current focal point to the next focal point.

Let's try out using three focal points as nodes to create a simple scene using a static camera.



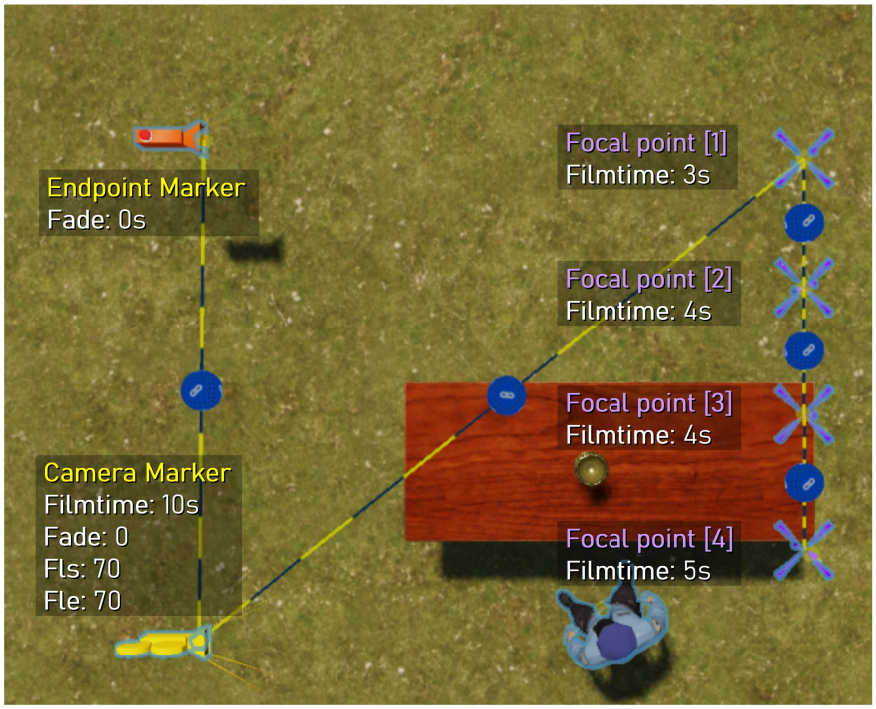
Once activated, our camera will point towards the first focal point. Immediately, it will begin to pan slowly to look at the second focal point. The first focal point has a **Filmtime** of 3 seconds, meaning it will take 3 seconds to pan to focal point 2.

As it reaches the second focal point, the camera will continue on without pausing, panning to look at the third and final focal point. We've instructed this leg of the journey, panning from focal point 2 to focal point 3, to be a little faster by setting focal point 2's **Filmtime** to 2 seconds. This means it will take 2 seconds to reach focal point 3.

Once looking at focal point 3, the camera will stop rotating. 5 seconds have now elapsed but, as the camera's own **Filmtime** is 7 seconds long, it will continue to hold on focal point 3 until the end of the shot.

We've set focal point 3's **Filmtime** to 0, but it doesn't really matter, as it's the last focal point in the chain.

Now let's create a more complex example. This time we're going to move the camera whilst changing the subject mid-shot.





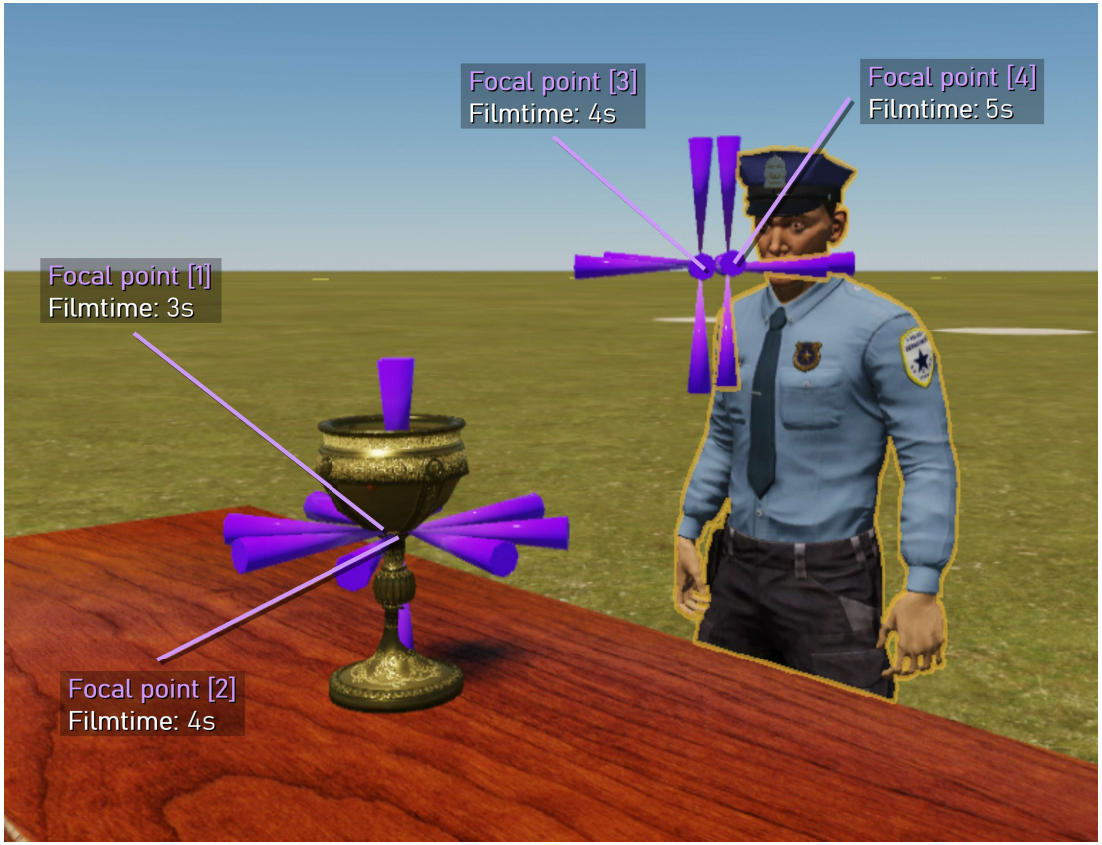
This shot is about revealing information; we start on a goblet, dolly past it and tilt up to reveal a police officer looking at it. We need to focus on the goblet for a bit, then move the camera's attention to the police officer, focusing on him for the remainder of the shot. This can be achieved using four focal points. Placing them in a line will make it easier to edit them and draw connections between them. Once we've connected them, we'll move them into position.

Focal points 1 and 2 will be placed very close to one another. Focal point 1's **Filmtime** is 3 seconds, meaning it will take 3 seconds to move the camera's focus to focal point 2. However, since focal point 2 will essentially be in the same location at focal point 1, the focus will effectively stay fixed on that spot – in this case, our goblet.

As the shot progresses, we've set focal point 2's **Filmtime** to 4 seconds, transitioning the focus from the goblet to focal point 3, which is placed just in front of the police officer. Remember focal points are invisible and have no collision so can be placed anywhere in the scene – including inside characters!

Once the camera's attention reaches focal point 3, it will look for focal point 4, which we've placed very close to focal point 3 to hold the focus in more or less in the same spot for the remainder of the shot. Focal point 4's **Filmtime** is the default, 5 seconds.

If you total up all the **Filmtimes** between all four focal points, you'll get 16 seconds, yet our camera is only set to be active for 10. Don't worry; the camera's **Filmtime** always takes precedence, so it doesn't matter if your focal point's **Filmtimes** don't add up. You could even use this creatively to cut to the next camera 'mid-pan', before the transition to the next focal point has finished.











The picture above gives a better impression of how these four focal points are arranged in the scene. The pictures on the right show what our finished shot looks like. This example can be found in the Cine Guru MAX examples fpm included in **Cine**|Guru **MAX**.

We encourage you to experiment with focal points – you can chain up to 30 of them together per-camera, allowing you to create complex sequences.

Focal targets

▶ **Video Tutorial:** https://youtu.be/XpbZt0ne2ec

Focal points will orient your cameras towards specific points in space, but tracking a moving object using them can be challenging. To solve this, we're going to make our moving object a "target" using a new marker exclusive to **Cine|Guru MAX** called the focal target.



Focal Target
Connecting a focal target to a camera and then to another entity makes the camera point towards that entity for the duration of its shot. By default, the camera will always point towards the entity's origin, this is not always its centre of mass. To overcome this, we can use the offset sliders to fine tune exactly where the camera points to.

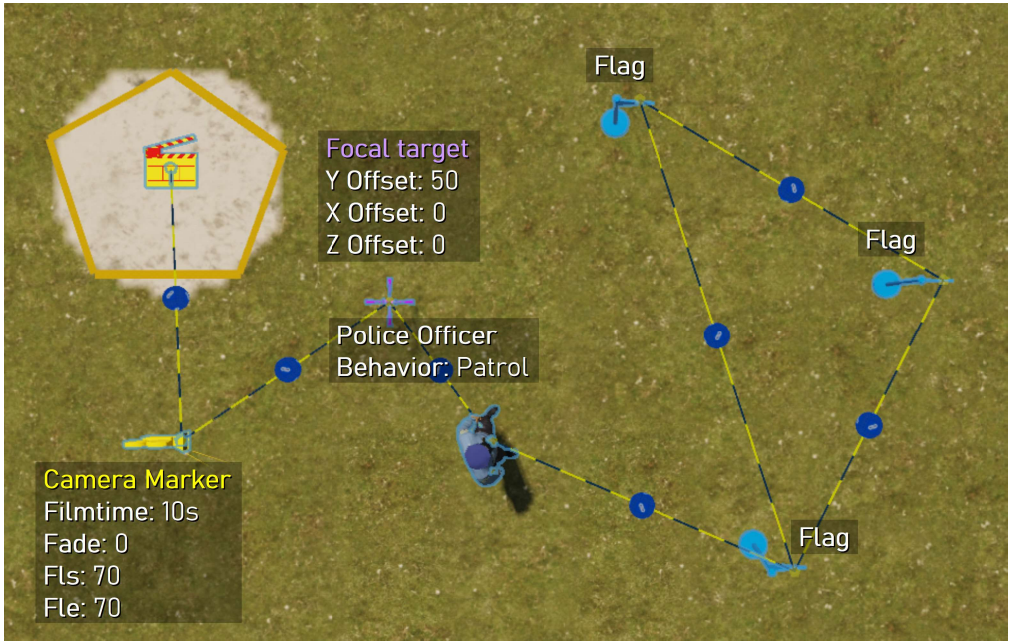


Y Offset | The vertical offset of the point of focus.

X Offset | The X offset of the point of focus.

Z Offset | The Z offset of the point of focus.

Here's a simple example using a static camera and a focal target attached to a moving entity – in this case, a character.



As the character patrols around the nearby flags, the camera will follow them. But by default, the camera would point at the character's feet because the origin of all characters in GameGuru MAX is the floor. So, we use the **Y offset** slider to raise this point of focus.

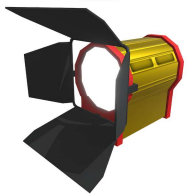
It's important to remember that the focal target itself does not point towards a specific point in space where the camera will focus, it only flags to the camera that the entity its connected to is what will be focussed on. You can place focal targets wherever you like, so long as they are connected to another entity.


LIGHTS

Light markers

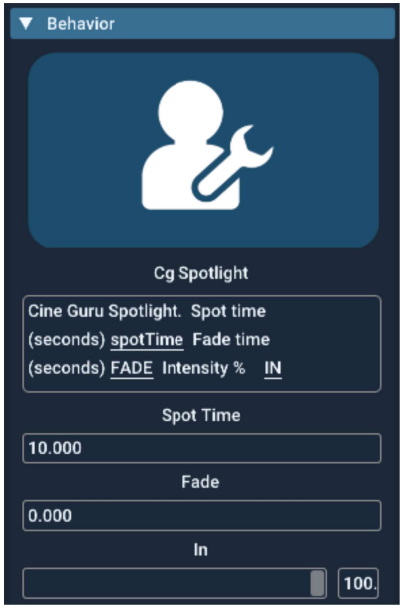
▶ **Video Tutorial:** https://youtu.be/68GpHKbQxck

New for **Cine**|Guru **MAX**, you can now use a set of markers to trigger and move dynamic lights in your scenes.



Light Marker

On its own, the light marker can be used to trigger a single light source (spotlight or point light) and control whether it fades in, how long it shines for, and at what intensity. The light marker has three editable parameters, which are:

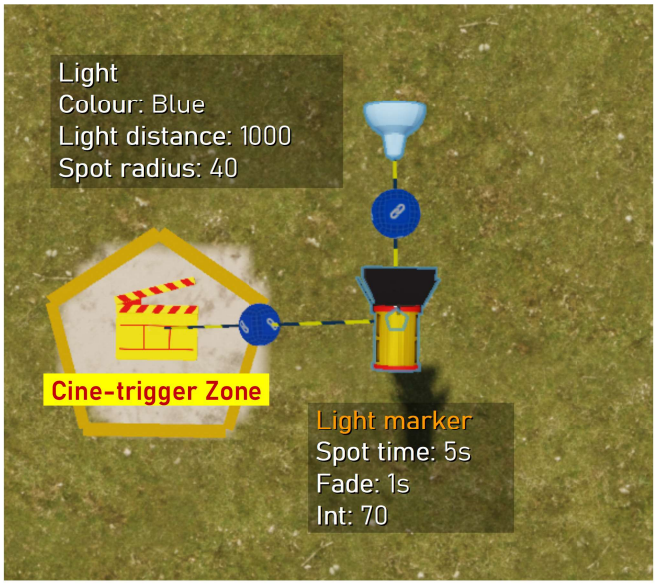


Spot time | The length of time the light is on, e.g. 5 seconds...
...or the length of time to travel to the next light node.

Fade | Specify a fade in for the light, e.g. 2 seconds.

Int | The intensity of the light, between 0-100%

Light markers do not produce light themselves, they must first be connected to a light via GameGuru MAX's visual logic system. The light marker inherits the colour, light distance and spotlight radius (if a spotlight) of the light it's connected to, the light itself inherits the position, orientation and – as we'll look at later – the direction of travel of the light marker.



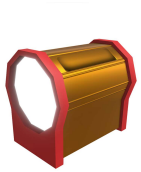
In this example, we've placed a light marker connected to a nearby spotlight. When we run the level, the spotlight won't switch on until the light marker it's connected to is triggered.

We've also connected the light marker to a cine-trigger zone so we can trigger it. You can activate light markers a few different ways which we'll look at later.

When the player enters the zone, the spotlight will be triggered. In the light marker, we've set the **spot time** to 5 seconds, meaning the spotlight will shine for 5 seconds, then switch off. It won't come straight on though, as we've also specified a 1 second **fade** in. We've also set the **int**ensity to 70% to dull the spotlight a little.

Light nodes

In the same way camera markers and camera nodes can create moving shots, light markers and light nodes can be used to move a light along a path.



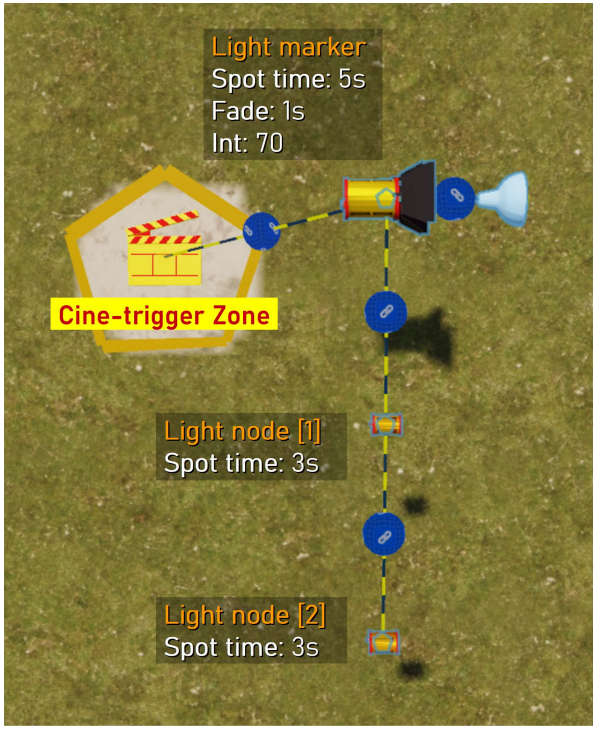
Light node
Connecting a light marker to a single light node creates a simple A-to-B path for a light to travel along, the travel time is dictated by the *spot time* of the light marker. Connecting a string of light nodes together creates a path the light will follow, with the *spot time* defining how long it takes to travel to the next light node.



Spot time | The length of time it takes the light to travel to the next node, e.g. 5 seconds...
...or, if it's the final node, the length of time to the light stays on.

Let's look at how we set up light nodes.

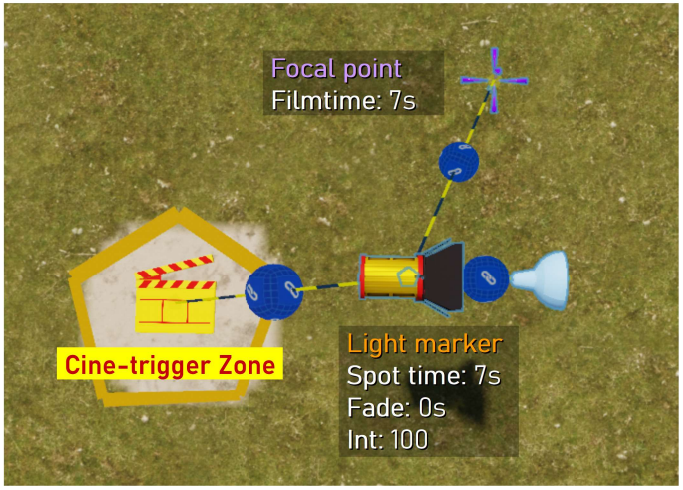
In this example, we've attached two light nodes to our light marker to create a path. The light marker's **spot time** is 5 seconds, meaning that – once activated – the light will take 5 seconds to reach the first node.



The first light node has a **spot time** of 4 seconds, meaning it will take 4 seconds for the light to travel to the final node.

The final node has a **spot time** of 3 seconds. Since there's no node after this one, the light will stop once it reaches the final node and the light will stay on for 3 seconds before switching off.

Note that while the light marker can control a fade in of the light, at the time of writing, there's no way to control a fade out. This option may be added in a future update of **Cine|Guru MAX.**





Using focal points with light markers

For greater control over where your light markers point, you can use focal points. This is most useful when using spotlights.

Using the Visual Logic Connections, connect your focal point to the light marker. Now, when activated, the light marker – and more importantly the spotlight attached to it – will point towards that focal point.

Just like with cameras, you can string multiple focal points together to create a "path of focus" for your spotlight. Remember, the focal point's **filmtime** parameters dictates how long it takes to transition the focus for the current focal point to the next focal point.

In this more complex example, our light is set to move "South" via two light nodes – taking 3 seconds to reach the first light node, and a further 4 to reach the last light node, whereupon the light will pause for 8 more seconds before deactivating.

At the same time, the light marker is connected to a string of focal points. This means that as well as physically moving, the direction it points will change too.

When activated, the spotlight will point at focal point [1], but it will immediately begin to swing "North" towards focal point [2], taking 3 seconds to get there. Upon reaching focal point [2], it will swing further "North" towards focal point [3], taking a further 4 seconds to get there. In this example, the *filmtime* of the final focal point ([3]) is technically irrelevant since there's no further instructions.

We encourage you to experiment with light markers and focal points to create unique lighting scenarios. Dynamic lighting is, after all, one of the key improvements of GameGuru MAX over its predecessor.

Cameras
Focus
Lights
Light markers
Light nodes
Using focal points with light markers
Activating light markers
Actors
Triggers
Preparing for standalone

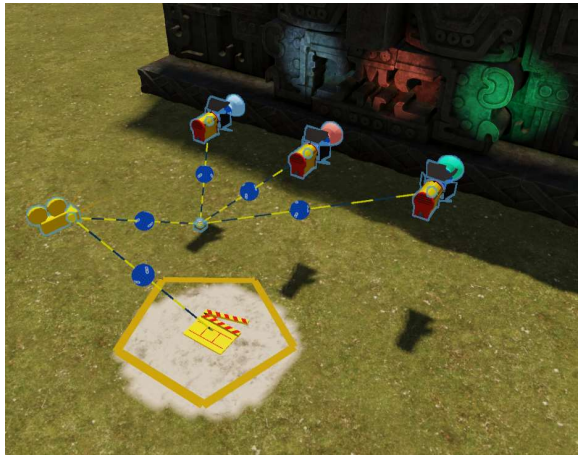
Activating light markers

There are three ways to activate light markers. Each method can be used to activate multiple light markers at once.



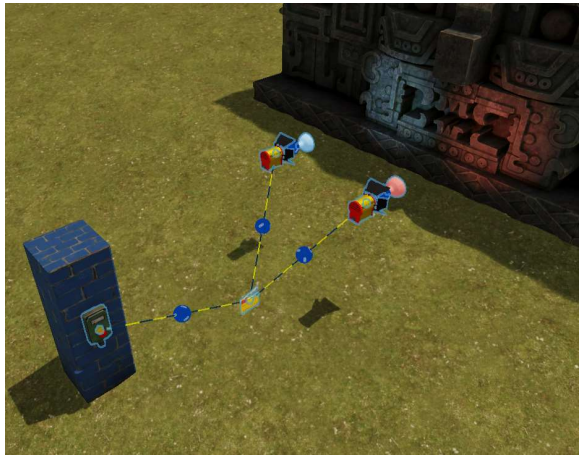
Cine-trigger zones
Cine-trigger zones can be used to activate light markers. Simply draw connections between the cine-trigger zone and the light markers you want to activate.

When the player enters the zone, the lights will immediately be activated.



Generic trigger (via a camera)
If you want to trigger your lights during a specific shot, connect a generic trigger to your chosen camera, then connect your lights to that generic trigger. We'll look at how to use generic triggers in the Triggers chapter.

Once this camera is active, the lights will be activated. This method also allows you to build a delay in before the activation.



Cine-trigger entity
Maybe you want the player themselves to activate your light markers. In which case, you can use a cine-trigger entity. Connect your switch or other activation method to a cine-trigger entity, then connect your light markers to the cine-trigger entity too.

Once the player activates the switch, this will activate the light markers.


ACTORS

Cameras
Focus
Lights
Actors
- **Actors**
- Marks
- Action text
- Looking up animations
- Writing action text
- Moving and talking
- Music commands
- Subtitles
- Activating and triggering
- Switching and aborting

Triggers
Preparing for standalone

Actors

▶ **Video Tutorial:** https://youtu.be/0mKZjwBBmZg

Cine|Guru MAX gives you the ability to control characters and objects during your cinematics, creating complex scripted events for actors. You can make them walk, talk, teleport, even vanish, as well as controlling music and subtitles. In this chapter we'll look at actors and how to control them.

First, some terminology to be familiar with:

Actor | Any entity that is being controlled by the script *cg_actor.lua* – this will usually be a character but it could be any dynamic entity.

Script | The script all actors need to run is *cg_actor.lua* – it uses the name of the actor to reference a text file which contains all of their actions.

Action text | A text file containing actions for the actors and other commands.



To set up an actor, place an entity, then in the *Object Tools* panel, give them a unique name – we've called this one "actor1" - and, in the *Behavior* pane, select *Cg Actor* as their behaviour.

Under *Developer Settings*, set them to *Always Active*. If you don't see a *Developer Settings* pane in the *Object Tools* panel, you'll need to enable this from GameGuru MAX's settings menu: (Edit > Settings > Developer > Display Object Tools Developer Mode).

When your entity is set to run the *Cg Actor* behaviour, you'll have the option to specify a text file and several sound files. This text file is where we will specify our action text, the instructions for our actor, the sound files will be our dialogue. As with all characters in GameGuru MAX, you can even record your own voice directly into GameGuru MAX here.

If you place down another actor, be sure to give them a unique name too, and specify their action text. Action texts can contain instructions for multiple actors.

Cameras
Focus
Lights
Actors
- Actors
- **Marks**
- Action text
- Looking up animations
- Writing action text
- Moving and talking
- Music commands
- Subtitles
- Activating and triggering
- Switching and aborting

Triggers
Preparing for standalone

Marks

Before we look at the action text itself, we should familiarise ourselves with actor's marks. Just like in real life, we can place down marks for where our actors should travel to. They work a bit like flags, except an actor will only travel to their next mark when prompted to, and will not return to a previous mark unless asked to. You can also make your actors look at marks (see "look" command, pg. 26)



Actor mark
T-shaped marks like this are common on real film sets, but we've added a slight point to ours to indicate which way your actor will face when they arrive at the mark. Rotate your mark to whatever direction you want your actor to face once at the mark. It's very important you give every mark in your level a unique name, e.g. "mark1", "mark2", mark3" etc. We'll cover their editable parameters later.



Bob

In order to better understand action texts, we're going to create a simple scene using two characters (a version of this scene is included with **Cine**|Guru **MAX** as CGM_Bob_and_Jane.fpm).

Both actors are running the *Cg Actor* behaviour and both are set to *Always Active*.

We need to give them unique names. Rather than calling them something generic like "actor1" and "actor2", let's called them "Bob" and "Jane".

Bob and Jane will both be controlled from the same action text. The action text will reference animations but, unlike GameGuru Classic, GameGuru MAX uses animation names rather than frame numbers. Make sure you give each animation a name when you create or import your character.



Jane

Action text

An action text is a text file containing instructions for our actors to follow, and a few other useful commands for our scene. Here's a list of all the commands you can use, along with examples of how to use them.

Command Example	**Description**
Actor commands	
idle|<animation name>|<animation speed> idle|Idle|0.9	Specifies the idle animation for the actor.
anim|<animation name>|<animation speed> anim|Zombie_Walk|1.1	Make actor play an animation.
move|<name of mark>|<speed>|<turn speed> move|mark1|50|50	Actor will walk to a specified mark.. Speed is a percentage of their normal walking speed; i.e. 100 is normal, 50 is half speed, 200 is double speed.
delay|<time in milliseconds> delay|2000	Actor will pause for a specified time, doing their idle animation whilst waiting.
look|<where to look> look|actor|jane look|mark|mark1 look|angle|120 look|forward look|player	Tells the actor to turn their head... - ...towards the actor named "jane" - ...towards the mark named "mark1" - ...towards a specific angle - ...straight forward - ...to look directly at the player ▶ **Look command:** https://youtu.be/Wr3kPv4ooTE
showwpn hidewpn showwpn hidewpn	If your character has a weapon, these two commands can be used to show and hide it on command.
teleport|<name of mark> teleport|mark1	Transport actor instantly to specified mark. To see a demonstration of the teleport command, check out the included *CG_teleport_and_switch.fpm*
speak|<sound slot>|<volume> speak|1|90	Play the sound in slot specified with specified volume.
sound|<sound slot>|<path to file> sound|1| audiobank\\characters\\human\\male\\enemyseesplayer5.wav	Swaps out one sound in the specified sound slot for another sound. This is often needed when using the switch and abort commands.

switch|<script name>
switch|character_attack

Causes actor to switch scripts.
Script must be specified in *cg_actor.lua* – we've included a few commonly using AI scripts you just need to comment in. To see a demonstration of the switch command, check out the included *CG_teleport_and_switch.fpm.*

activate
activate

Activates whatever entity is specified in actors IfUsed field.

trigger
trigger|camera1

Triggers the specified camera

vanish
vanish

Causes actor to be destroyed. No other commands can be called after this.

Music commands

music, <path to audio file>, <length in milliseconds>
music, audiobank\\music\\outside.ogg, 29536

Specifies a music track.

mplay|<fade in time in ms>
mplay|2000

Plays music with a fade in. The track will loop, and override any existing music track playing.

mstop|<fade out time in ms>
mstop|5000

Fades out the music then stops it.

mvol|<music volume>|<change time in ms>
mvol|70|2500

Change the volume of the music currently playing.

Subtitle commands

subtitles, <filepath to text file>,
<Y location of first line of subtitles in screen %>,
<Y spacing of lines in screeen %>,
<Size of text to use, i.e. the size value to go in GG's Text Lua commands>
subtitles, scriptbank\\Cine Guru MAX\\subtitles.txt, 80, 4, 4

Tells *cg_actor.lua* where to find the subtitles text file.
Where to display the subtitles.
How far apart the lines should be.
What font size to display them in.

sub|<dialog line>
|<time to display>
|<subtitle line>
|<width of backdrop in screen %>
sub|3|4000|2|20

Which line of the subtitles file should be displayed?
How long should it be displayed?
Should it display on line 1 or 2?
How wide should the black strip behind this line be?

abort,<actor name>,<another command>
abort,actor1,switch|character_attack

Specifies what should happen if the cinematic is aborted (Spacebar pressed for more than 0.25 secs)

debug
debug

Displays some prompts for user info. Useful for working out subtitle backdrop percentages.

Cameras
Focus
Lights
Actors
- Actors
- Marks
- Action text
- **Looking up animations**
- Writing action text
- Moving and talking
- Music commands
- Subtitles
- Activating and triggering
- Switching and aborting

Triggers
Preparing for standalone

Looking up animations

Whether you're using stock GameGuru MAX characters, ones you've made in MAX's character creator, or entirely custom ones, if you're planning on using them as actors, at some point you'll need to reference their animations by name. Unfortunately, at the time of writing, MAX doesn't make this information easily accessible, so here's a tip to looking up what animations your character contains.



1 Place a copy of your actor somewhere in the level. This is our “dummy”.



2 Assign it the behavior: *Play Animation.lua.*



3 You will now be able to see a list of animations in the drop-downs.



4 If you want to preview an animation, set the *Action Play* to whatever animation you'd like to see.



5 In test game, run over to your “dummy” character and watch the animation play when you get near.

```
jane, move|mark2, delay|2000

clint, anim|Anim_Unequip_to_Idle, de

jane, speak|2|75, teleport|mark3
```

6 If you like the animation, make a note of its name. You can then reference this animation in your action text.

Cameras
Focus
Lights
Actors
- Actors
- Marks
- Action text
- Looking up animations
- **Writing action text**
- Moving and talking
- Music commands
- Subtitles
- Activating and triggering
- Switching and aborting

Triggers
Preparing for standalone

Writing action text

▶ **Video Tutorial:** https://youtu.be/YHpNIo62FCE

Now we'll write an action text to control our characters Bob and Jane. Every new addition is highlighted in green.

1 Create a new blank text file and call it "actions.txt". Open it and type in the actor's name, followed by a comma, then write your first command. The idle command is a good one to start with; the actor will revert to this animation when not doing anything else. We will name our idle animation and specify a speed.

```
Jane,idle|Idle_0.9
```

2 We'll define Bob's idle animation too. If you don't specify an animation speed, it will default to 1.

```
Jane,idle|Idle_0.9
Bob,idle|Idle_LookAround
```

3 To animate a character, use the anim command and call your animation by name.

```
Jane,idle|Idle_0.9
Bob,idle|Idle_LookAround
Jane,anim|Throw_Grenade
```

4 We'll give Bob an animation too. Many of the commands in action text are designed to run sequentially, not concurrently. The anim command is one such example; Bob will only play his animation once Jane has finished hers.

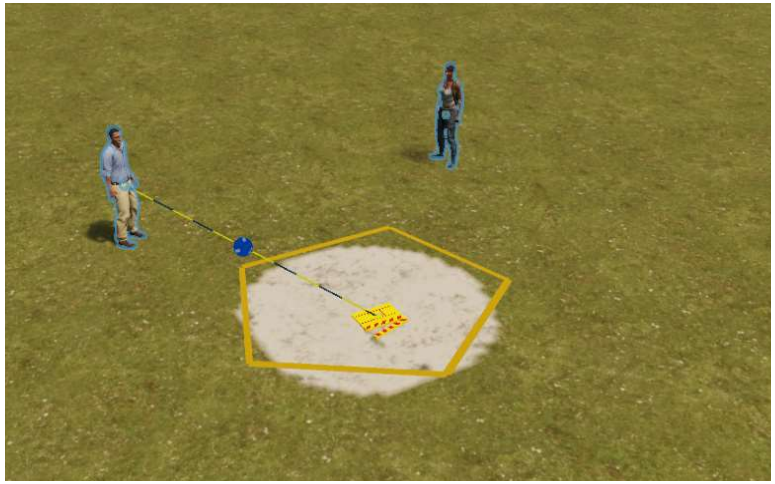
```
Jane,idle|Idle_0.9
Bob,idle|Idle_LookAround
Jane,anim|Throw_Grenade
Bob,anim|Throw_Grenade
```

5 We can add a slight pause before Bob performs his animation by adding the delay command before his anim command.

```
Jane,idle|Idle_0.9
Bob,idle|Idle_LookAround
Jane,anim|Throw_Grenade
Bob,delay|1000,anim|Throw_Grenade
```

Save your action text. Back in GameGuru MAX, we'll add a cine-trigger zone use the visual logic system to connect it to one of our actors (it doesn't matter which one).

When the player enters this zone, it will trigger our actors to play out their actions as specified in the action text.



Cameras

Focus

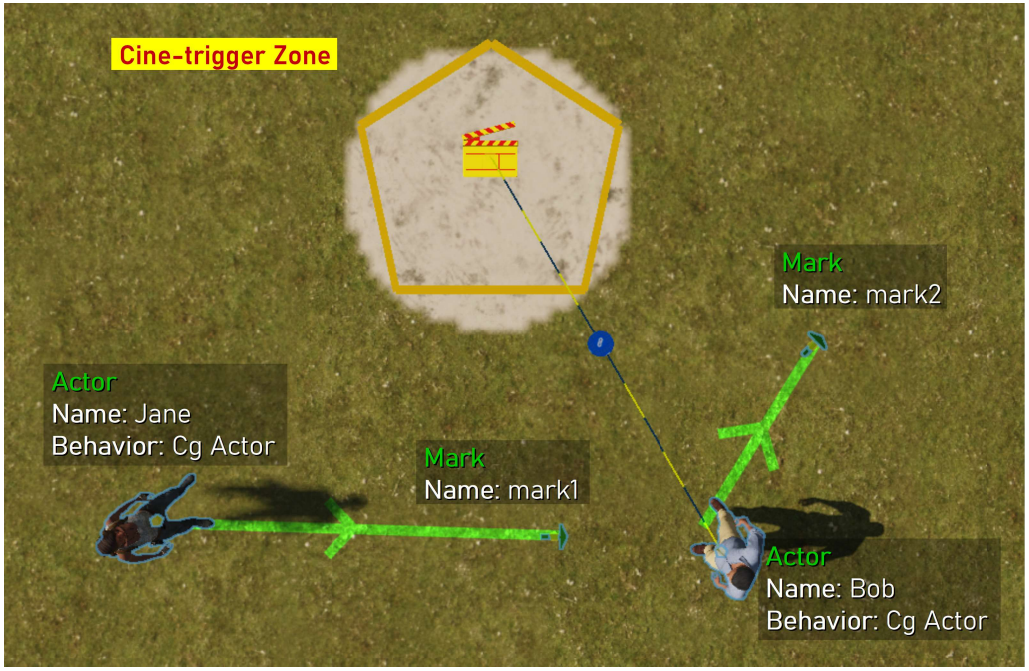
Lights

Actors

- Actors
- Marks
- Action text
- Looking up animations
- Writing action text
- **Moving and talking**
- Music commands
- Subtitles
- Activating and triggering
- Switching and aborting

Triggers

Preparing for standalone



Moving and talking

Now we'll add some marks for our actors to move to. Place down two marks and give them unique names – “mark1” and “mark2”.

Let's change up the sequence slightly too. Jane is going to walk into the scene up to “mark1”. They will have a short conversation, then Bob is going to walk to “mark2”.

This diagram (left) shows how we've set our scene up, including the visual logic connection.

6 Use the move command to instruct Jane to walk to “mark1”. Once there, we'll instruct Bob to walk to “mark2” also using the move command.

```
Jane,idle|Idle_0.9
Bob,idle|Idle_LookAround
Jane,move|mark1
Bob,move|mark2
```

We can use the speak command to make our actors talk. We'll assume we've already recorded two lines of dialogue per actor. Assign these sounds to the sound slots of each respective actor, GameGuru MAX will automatically generate a lip-sync file. We can then call these in as needed and set their volume.



7 Insert the speak command, specifying which slot should be played when, and at what volume. It's important you put the speak command before a related talking animation so that it plays over the animation and not after it. GameGuru MAX doesn't ship with talking animations as standard, but these can be added to characters if you purchase the Animation Booster Pack DLC. If don't have this, you can use the delay command instead.

```
Jane,idle|Idle_0.9
Bob,idle|Idle_LookAround
Jane,move|mark1,speak|0|90,delay|6000
Bob,speak|0|94,delay|3000
Jane,speak|1|90,delay|3000
Bob,move|mark2,speak|1|91
```

Music commands

▶ **Video Tutorial:** https://youtu.be/Jw7hnifdsvY

Now we will add music to our scene, and control its volume using the action text.

8 Use the music command to specify your music track, it's best to do this at the beginning of your action text. You also need to specify the length of the track in milliseconds in order for it to loop correctly. You can use a DAW (digital audio workstation) like Audacity to find out the exact length of your track.

```
music,audiobank\\music\\outside.ogg,29536
Jane,idle|Idle_0.9
Bob,idle|Idle_LookAround
Jane,move|mark1,speak|0|90,delay|6000
Bob,speak|0|94,delay|3000
Jane,speak|1|90,delay|3000
Bob,move|mark2,speak|1|91
```

9 We want our music to play right away, so we'll use the mplay command. We'll also specify 3 second fade in on the track.

```
music,audiobank\\music\\outside.ogg,29536
Jane,mplay|3000,idle|Idle_0.9
Bob,idle|Idle_LookAround
Jane,move|mark1,speak|0|90,delay|6000
Bob,speak|0|94,delay|3000
Jane,speak|1|90,delay|3000
Bob,move|mark2,speak|1|91
```

10 By default, music is played at 100% volume, which may be too loud to hear our characters talk. Here's we'll use the mvol command to gradually lower the volume. The command mvol|70|2000 will lower the volume to 70% over the course of 2 seconds.

```
music,audiobank\\music\\outside.ogg,29536
Jane,mplay|3000,idle|Idle_0.9
Bob,idle|Idle_LookAround
Jane,move|mark1,mvol|70|2000,speak|0|90,delay|6000
Bob,speak|0|94,delay|3000
Jane,speak|1|90,delay|3000
Bob,move|mark2,speak|1|91
```

11 Finally, we'll fade out and stop the music by using the mstop command. The number after mstop| defines the length of the fade out in milliseconds. We've also added a small delay before mstop so the music doesn't immediately fade out after Bob has finished animating.

```
music,audiobank\\music\\outside.ogg,29536
Jane,mplay|3000,idle|Idle_0.9
Bob,idle|Idle_LookAround
Jane,move|mark1,mvol|70|2000,speak|0|90,delay|6000
Bob,speak|0|94,delay|3000
Jane,speak|1|90,delay|3000
Bob,move|mark2,speak|1|91,delay|2000,mstop|3000
```

Cameras
Focus
Lights
Actors
- Actors
- Marks
- Action text
- Looking up animations
- Writing action text
- Moving and talking
- Music commands
- **Subtitles**
- Activating and triggering
- Switching and aborting

Triggers
Preparing for standalone

Subtitles

▶ **Video Tutorial:** https://youtu.be/4PM9Fjkm0KA

You can add subtitles to your cinematics, controlled via the action text. The subtitles themselves are stored in a separate text file referenced by the action text. We'll create a new text file called *subs.txt*. The picture, right, shows how you might lay out a subtitle text.

```
[Tense music]
Jane: General, there's a problem in sector 34.
The main door won't unlock for any personnel.
Bob: That's a door down the hall isn't it?
Jane: Yes, the big green shutter door.
Bob: Luckily for you, I can open it from here.
```

12 Use the subtitles command specify the text file containing the subtitles. The three numbers after the filename are: the Y value (in screen percentage) of the top line of subtitles (80%), the spacing between the two lines of subtitles (4%), and the font size to display the subtitles in (font size 4). GameGuru's text doesn't compensate for screen resolution; the higher the res., the smaller the text will appear, so choose your font size wisely. We'll also add a debug command for later.

```
music,audiobank\\music\\outside.ogg,29536
subtitles,scriptbank\\Cine Guru MAX\\subs.txt,80,4,4
debug
Jane,mplay|3000,idle|Idle_0.9
Bob,idle|Idle_LookAround
Jane,move|mark1,mvol|70|2000,speak|0|90,delay|6000
Bob,speak|0|94,delay|3000
Jane,speak|1|90,delay|3000
Bob,move|mark2,speak|1|91,mstop|3000
```

13 The sub command displays a subtitle. It has four parameters; 1) which line of the subtitle text we want to display – here, it's line one. 2) How many milliseconds the subtitle should be displayed for. 3) Which line the subtitle should be displayed on – line 1 or line 2. Generally, single line subtitles should be displayed on line 2 as it's less intrusive. We'll leave the fourth parameter, backdrop width, blank for now.

```
music,audiobank\\music\\outside.ogg,29536
subtitles,scriptbank\\Cine Guru MAX\\subs.txt,80,4,4
debug
Jane,mplay|3000,idle|Idle_0.9,sub|1|3000|2
Bob,idle|Idle_LookAround
Jane,move|mark1,mvol|70|2000,speak|0|90,delay|6000
Bob,speak|0|94,delay|3000
Jane,speak|1|90,delay|3000
Bob,move|mark2,speak|1|91,mstop|3000
```

The debug command will display what state your actors are in but also, when used in conjunction with a sub command, it will display how wide the black backdrop behind each subtitle is. If you leave the fourth parameter of a sub command blank, the script guesses based on the font size you chose and the number of characters in the line being displayed. This estimate is shown at the bottom of the screen. Use this number to refine the actual backdrop width needed – it's usually an overestimate. In our example, right, the script has estimated a backdrop 13% of screen width. This is slightly too wide, so we will drop it to 11.
Our sub command for this line will now read: sub|1|3000|2|11.



14 Use a combination of the sub and debug commands to work out the backdrop width needed for each subtitle. We usually want to place subtitles immediately after speak commands so the lines display exactly when each actor is talking.

This step can initially be a lot of trial and error but, with practice, you'll get better at estimating backdrop widths yourself.

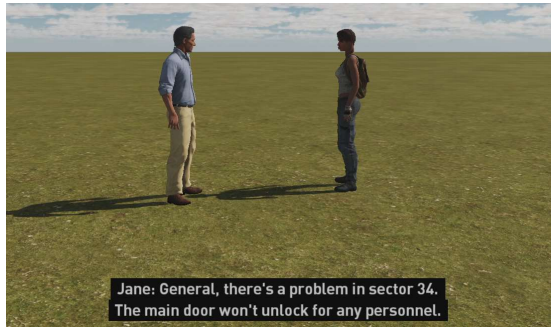
```
music,audiobank\\music\\outside.ogg,29536
subtitles,scriptbank\\Cine Guru MAX\\subs.txt,80,4,4
debug
Jane,mplay|3000,idle|Idle_0.9,sub|1|3000|2|11
Bob,idle|Idle_LookAround
Jane,move|mark1,mvol|70|2000,speak|0|90,sub|2|6000|
1|35,sub|3|6000|2|35.5,delay|6000
Bob,speak|0|94,sub|4|3000|2|30,delay|3000
Jane,speak|1|90,sub|5|3500|2|28,delay|3000
Bob,move|mark2,speak|1|91,sub|6|5000|2|34,delay|
3000,sub|7|3000|2|14,mstop|3000
```

15 Finally, you can comment out any line by adding -- at the start. In this case, we'll comment out our debug command. Note that this only works at the beginning of a line, not mid-way through it or on individual commands.

```
music,audiobank\\music\\outside.ogg,29536
subtitles,scriptbank\\Cine Guru MAX\\subs.txt,80,4,4
--debug
Jane,mplay|3000,idle|Idle_0.9,sub|1|3000|2|11
Bob,idle|Idle_LookAround
Jane,move|mark1,mvol|70|2000,speak|0|90,sub|2|6000|
1|35,sub|3|6000|2|35.5,delay|6000
Bob,speak|0|94,sub|4|3000|2|30,delay|3000
Jane,speak|1|90,sub|5|3500|2|28,delay|3000
Bob,move|mark2,speak|1|91,sub|6|5000|2|34,delay|
3000,sub|7|3000|2|14,mstop|3000
```

We've now created a simple exchange between two characters, with triggered dialogue, controlled music, and subtitles.













Cameras
Focus
Lights
Actors
- Actors
- Marks
- Action text
- Looking up animations
- Writing action text
- Moving and talking
- Music commands
- Subtitles
- **Activating and triggering**
- Switching and aborting

Triggers
Preparing for standalone

Activating and triggering

▶ **Video Tutorial:** https://youtu.be/ret_KLFLj6I

Now we're going to use the activate and trigger commands to open a door and initiate a camera. We'll also have Jane walk through the previously shut door.

To do this, we'll need to place a door and a new actor mark down, but depending on your level, you might also need to place additional scenery down to compliment GameGuru MAX's AI pathfinding logic. In some cases, if the route to a mark is blocked at the start of the cinematic, the whole cinematic won't play, or the actor will go the long way round. Here, we've enclosed the mark in its own room, meaning Jane has no option but to go through the door. Needless to say, this can take some trial and error.



In order for Bob to activate this door, we need to put the name of the door in his IfUsed field. This can be found in the Developer Settings panel.

16 Add the activate command (followed by a comma) for an actor to activate whatever is named in their IfUsed field.

```
Bob,move|mark2,speak|1|91,sub|6|5000|
2|34,delay|3000,activate,
```

We discovered with the original **Cine|Guru**, it was useful to have actors be able to trigger cameras themselves, especially during longer cinematics where fluctuations in framerate can cause cameras and actions to go out of sync. To trigger a camera, give the camera marker a unique name, then use the trigger command in the action text, followed by the name of the camera.

17 Add the trigger command, followed by the name of the camera marker.

```
Jane,trigger|camera1,move|mark3,sub|7|
3000|2|14,mstop|3000
```

Now, as Jane sets off to mark3, the cinematic takes control of the camera to show her walking through the now open door. Cameras triggered from action texts override any camera that is already running.

Cameras
Focus
Lights
Actors
- Actors
- Marks
- Action text
- Looking up animations
- Writing action text
- Moving and talking
- Music commands
- Subtitles
- Activating and triggering
- **Switching and aborting**

Triggers
Preparing for standalone

Switching and aborting

▶ **Switching and teleporting:** https://youtu.be/nnVduKDoXe0

At the end of a cinematic, you may wish to switch the behaviour of an actor to something more useful in-game, like that of an enemy – otherwise they will keep playing their idle animation indefinitely. Let's look at how you can achieve this.

1 Open *cg_actor.lua*. This can be done in Notepad, but we recommend using Notepad++ for editing and saving Lua scripts as it is more user-friendly.



2 At the top of script, you will see a number of commented out lines with some of the common stock AI scripts. Simply delete the -- to comment them back in. In Notepad++, the word "require" will turn blue.

```
-- For instructions on use see Cine Guru documentatio
-- Commented out 'require' lines for use with actor s
--:
-- require "scriptbank\\people\\character_attack"
-- require "scriptbank\\people\\zombie_attack"
-- require "scriptbank\\people\\patrol"
--:
local lower = string.lower
```

3 You can also do this with your own AI scripts. Simply reference them in the same way, using the require command, followed by the filepath. Note you don't need to include the file extension (.lua) here.

```
-- For instructions on use see Cine Guru documentatio
-- Commented out 'require' lines for use with actor s
--:
require "scriptbank\\people\\character_attack"
require "scriptbank\\people\\my_awesome_behaviour"
-- require "scriptbank\\people\\zombie_attack"
-- require "scriptbank\\people\\patrol"
--:
```

4 Now in your action text, use the switch command to tell the actor what script to switch to and when. Again, don't include the .lua file extension.

```
Bob,move|mark2,switch|character_attack
Jane,move|mark3,switch|patrol
```

5 Because of how GameGuru MAX behaviours deal with sounds, we may need to swap back in the original sounds we previously swapped out for our custom dialogue sounds. If we don't, when you actor switches behaviour, they'll play dialogue sounds instead of their default sounds. To make sure this doesn't happen, use the sound command and specify each file that needs to be swapped back in, and into which sound slot.

```
Bob,sound|0|
audiobank\\characters\\human\\male\\en
emies_heard_something3.wav,
Bob,sound|1|
audiobank\\characters\\human\\male\\en
emyseesplayer5.wav,
Bob,move|mark2,switch|character_attack
```

6 The abort command specifies what an actor should do if the player chooses to skip or abort the cinematic (by holding the space bar down for half a second). It's best practice to put your abort commands in at the beginning of your action text.

```
abort,Bob,vanish
abort,Jane,switch|sniper4
abort,Tim,teleport|mark5
```

▶ **Aborting actors:** https://youtu.be/vLdB9FZkDkI


TRIGGERS

Cine-trigger zones

▶ **Video Tutorial:** https://youtu.be/MBWh6dfmiPE

In this chapter, we'll cover the various triggers that come with **Cine|Guru MAX** and how to get the most out of them, starting with the most important one; the cine-trigger zone.





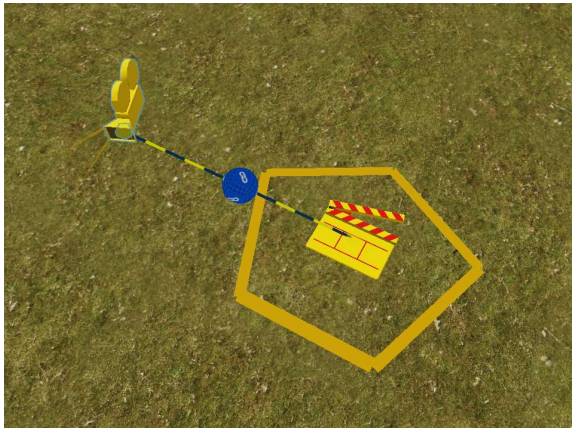
Cine-trigger zone
Because cine-trigger zones are zones rather than entities, they don't appear in your Level objects list on the left side of the GameGuru MAX editor. Each time you want to add one to your level, you will need to navigate to it within your Cine Guru MAX folder in entitybank. Cine-trigger zones have a couple of editable properties to be aware of.

Dtime | Specify delay before your cinematic is triggered.

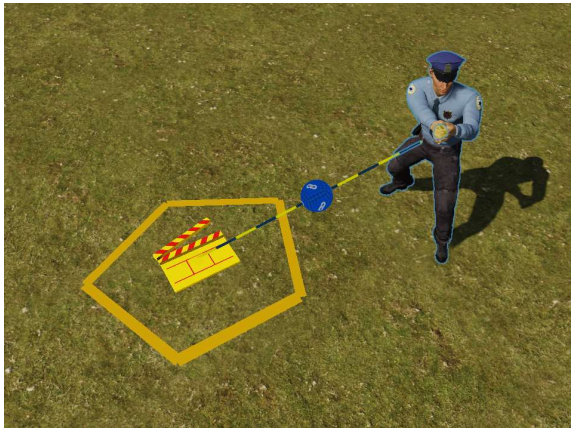
Enable Skip | If ticked, this option allows the player to skip a cinematic by holding the space bar for 0.25 seconds. This only works if there is a camera active.

Disable If Combat | If ticked, the cinematic won't trigger if the player is under fire.

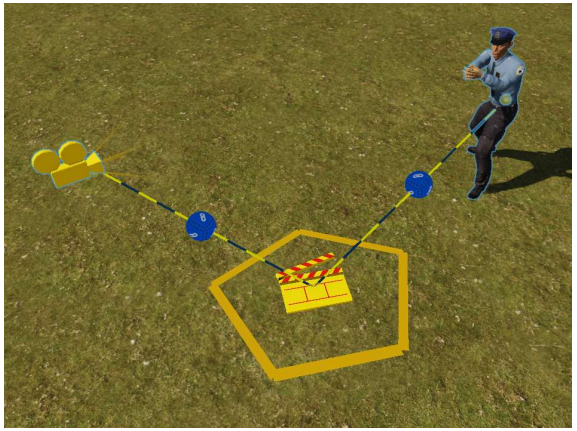
Use GameGuru MAX's Visual Logic system to connect a cine-trigger zone to different markers to achieve different results.



Trigger a camera or sequence of cameras by connecting the cine-trigger zone to the first camera in the sequence.



Trigger an actor by connecting the cine-trigger zone to an actor. All actors that share the same action text will be activated.



A single cine-trigger zone can trigger both an actor and a camera at the same time.

Cine-trigger entities

▶ **Video Tutorial:** https://youtu.be/Hpk-8TVMqIE

Like cine-trigger zones, cine-trigger entities can activate cameras, actors or both. Whereas zones require the player to enter a designated area to trigger, cine-trigger entities can be triggered in a variety of ways, including by player interaction.



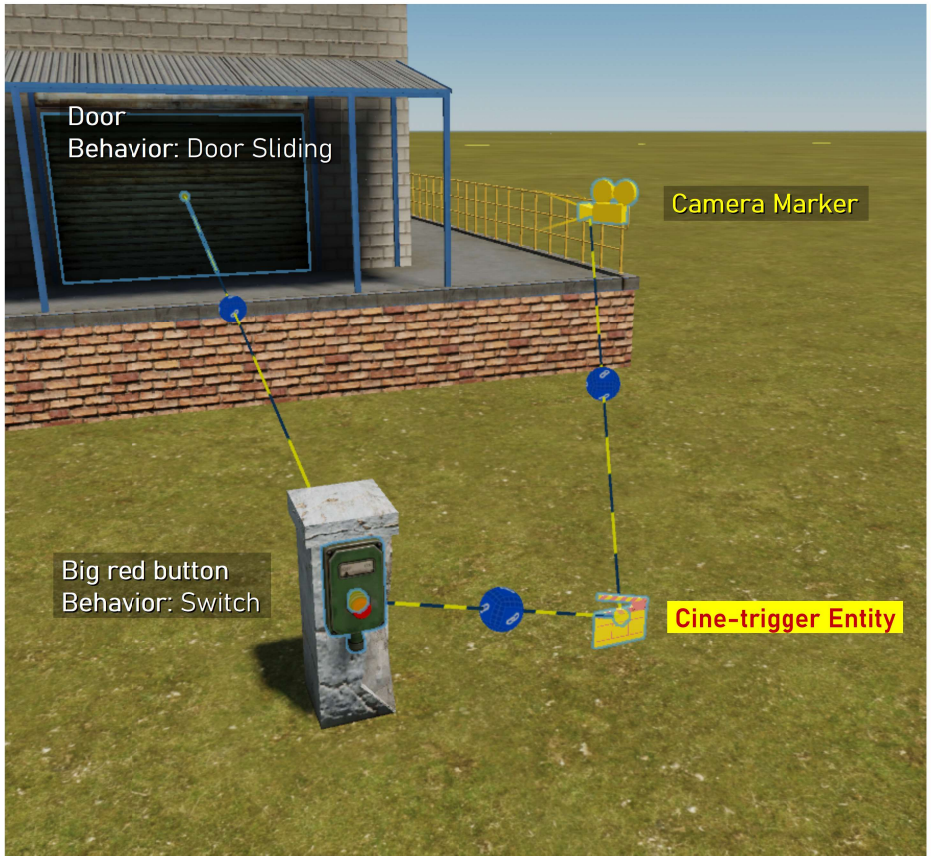
Cine-trigger entities
Simply drop a cine-trigger entity into your level as you would any other entity, then use the visual logic system to connect it to an element or elements you wish to trigger. It has the same editable properties as a cine-trigger zone (**Dtime**, **Enable Skip**, and **Disable If Combat**).

Example: cine-trigger entities in action
GameGuru MAX's Visual Logic Connections make the relationships between the entities easier to understand.

The big red button is running the "switch" behaviour; this will activate whatever entity it is connected to. In this example, it's connected to a door and our cine-trigger entity. When the player approaches the big red button and presses 'E', the door will open. Our cine-trigger entity will also be triggered.

By triggered the cine-trigger entity, any elements connected to it will also be triggered. In this example, we have a simple static camera attached to the cine-trigger entity, so when the cine-trigger entity is activated, so too is that camera.

In practice, all this simply means that when the player presses E at the big red button, their view cuts to that of our camera to show the door opening. You can imagine how this setup could be used to reveal new information to the player quickly and easily.

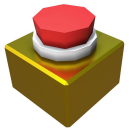


Generic triggers

▶ **Video Tutorial:** https://youtu.be/iFIsXlWNh-w

During your cinematics, you may want to trigger additional actions. **Cine|Guru MAX** comes with a simple but powerful all-purpose trigger to do just that. It even has a built-in delay timer you can set.





Trigger
Connect a trigger to the camera you want it to activate during, then use the built-in delay parameter to offset the triggering. For example, set it to 3.000 to trigger three seconds into your camera's shot.

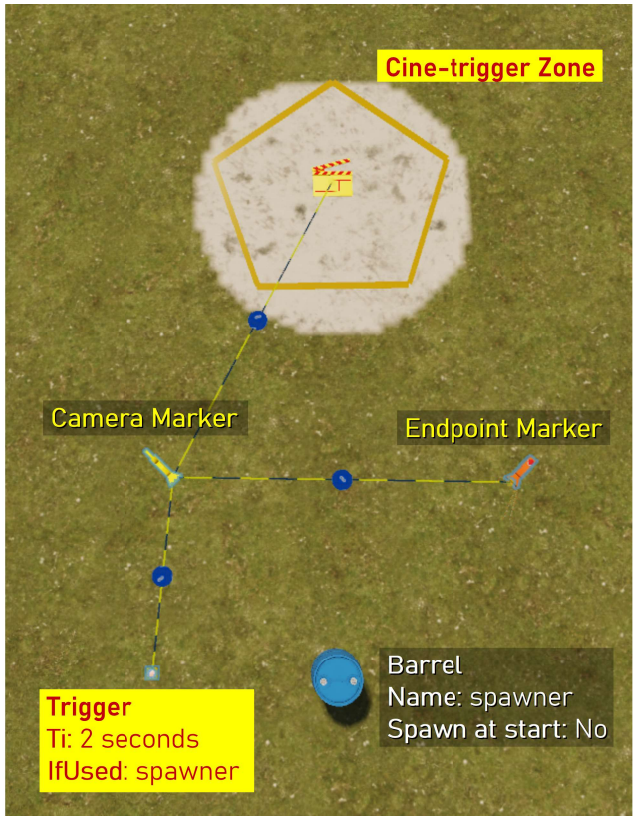
Ti | The delay before triggering, e.g. 5.000 = 5 seconds.

A simple way to demonstrate triggers is to spawn an entity during a shot, as in this example.

1 In this simple scene, we've set up a cine-trigger zone to trigger a camera, which travels to an endpoint.

2 We've also placed a barrel into the scene and, in its Developer Settings, unticked the box *Spawn At Start?* This means the barrel won't exist until we activate it. We've given the barrel a unique name; "spawner", which our trigger will reference later.

3 Attached to our camera marker is a generic trigger. This means that when the camera is active, the trigger will activate after whatever delay we've set. In this case we've set a 2 second delay before activation.



4 Triggers work by activating whatever entity is named within their IfUsed field. In this case, we've entered the unique name of the barrel, "spawner".

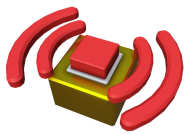
5 When the player enters the cine-trigger zone, our camera will be activated and, after a delay of 2 seconds, so too will our trigger, which will in turn activate the "spawner" barrel, calling it into existence.

! Remember, you can link multiple triggers to the same camera if you want to trigger multiple elements during the same shot.

Camera shake triggers

▶ **Video Tutorial:** https://youtu.be/UKxC4EdjI4k

So far, all the camera moves we've looked at have been either locked off like they are on a tripod, or buttery smooth like they are on a dolly, jib, or simply floating in space. The camera shake trigger is a way to add a more grounded feel to your shots, simulating a handheld camera, or emphasising big moments of impact.



Camera shake trigger
Connect a camera shake trigger to the camera you want to add some shaking too. It has two modes of operation; Set and Add.
"Set" shakes the camera for the entire duration of the shot once triggered.



Mode | **Set**

Ftime | The delay before triggering, e.g. 0.500 = half a second.

Trauma | The severity of the shaking (0-100%).

Period | The speed of the shaking – small values are fast, large values are slow.

Fade | Does nothing when the mode is set to "Set".

When the mode of the camera shake trigger is set to "Add", it shakes the camera after the specified delay, then subsides to nothing. "Add" can be used to shake the camera mid-way through a shot, perhaps to create an earthquake effect or emphasise an explosion.

Mode | **Add**

Ftime | The delay before triggering, e.g. 0.500 = half a second.

Trauma | The severity of the shaking (0-100%).

Period | The speed of the shaking – small values are fast, large values are slow.

Fade | How quickly the shaking subsides. Small values deduct a small amount of shaking per frame and create a longer tail off, large values reduce the shaking quickly.

Let's look at some scenarios where you could use camera shake triggers.

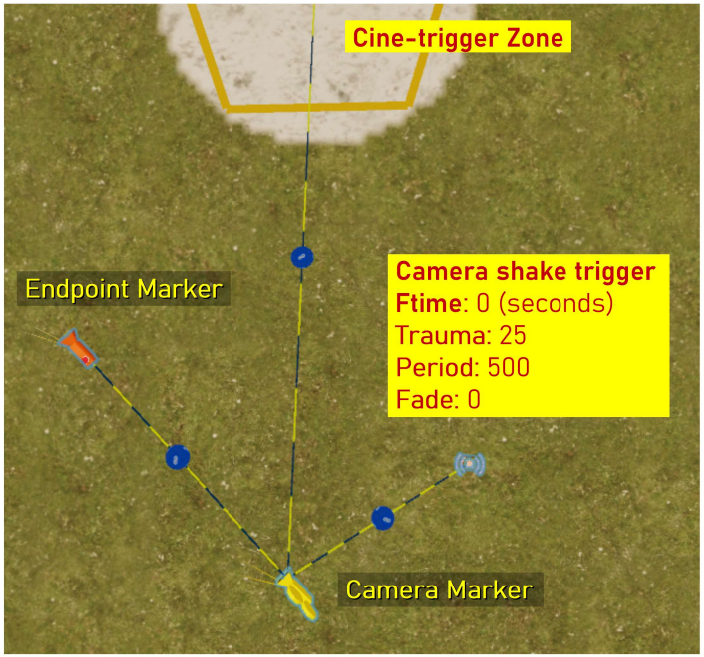
Set

In this example, we have a simple A-to-B camera move where the camera pushes forward. We can use a camera shake trigger to make this shot look like it was shot with a handheld camera, or is a character's point-of-view.

Using the visual logic system, we've connected a camera shake trigger to our camera marker. We've set the mode of the camera shake trigger to "Set" and used the following values to simulate a handheld camera:

Ftime = 0, **Trauma** = 25, **Period** = 500, **Fade** = 0 (does nothing)

When this camera is active, the shaking will start and last the duration of the shot.



Add

Here we've set up other simple camera move, pulling back from a red barrel. The barrel itself is running a simple script to explode when triggered, and we've dropped in a generic trigger to control this, which is attached to our camera. Two seconds into the shot, the barrel is triggered to explode.

Also connected to our camera marker is a camera shake trigger set to "Add". We've used an **Ftime** of 2, to delay the start of the shaking until two seconds into the shot, this should match exactly with our barrel exploding.

To make the explosion appear more violent, we've used the maximum **Trauma** value of 100, and a short **Period** to shake the camera quickly. The **Fade** is set to 3; this means the intensity of the shake decreases by 3% every frame. Simply maths tells us it will take 34 frames to reduce the shaking from 100 to 0, or just over half a second. If we wanted a more gradual fade, we could use a smaller value. At 60fps, a **Fade** of 0.1 would take 1000 frames, or 16.7 seconds to subside to nothing.

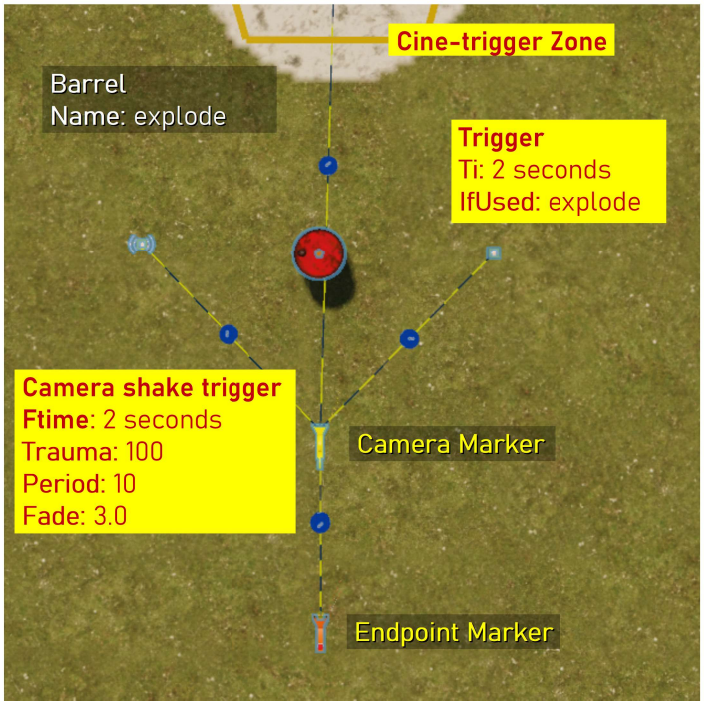


Image triggers

▶ **Video Tutorial:** https://youtu.be/ffRbbew-6l0

Image triggers can be used to display sprites on-screen. Unlike the image triggers in the original **Cine|Guru**, which were based on the image zones, GameGuru MAX's dynamic lua capabilities allowed us to make these triggers entities.



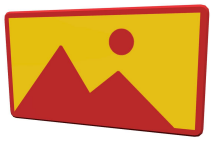
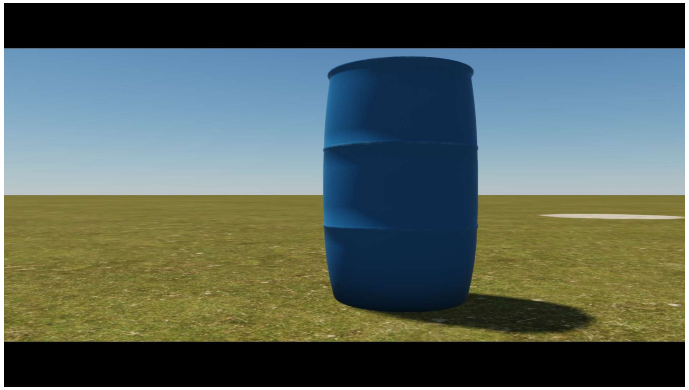


Image Trigger
Drop an image trigger into your level and connect it to the camera you'd like to display your sprite during. Image triggers have three modes, not all parameters are needed for all modes.

Image Type | Can be either:	**Letterbox**	**Image**	**Sequence**
Image File |	(not needed)	Select your image file	First image in sequence
Trigger Time |	(does nothing)	Delay before displaying	Delay before displaying
Image Duration |	(does nothing)	How long to display	Total sequence length
Frame Time |	(does nothing)	(does nothing)	How many milliseconds per frame?

Let's look at the three modes in more detail and how to use them.



Letterbox
Displays an image over all cinematics in a level.

Whenever there's a camera active, this image will display on screen, no other parameters are needed. It's designed for adding letterbox bars to your cinematics, we've included an example letterbox as part of this pack.

You don't need to specify an **Image File**, **Trigger Time**, **Image Duration**, or **Frame Time** when using **Letterbox**.

Cameras
Focus
Lights
Actors
Triggers
Cine-trigger zones
Cine-trigger entities
Generic triggers
Camera shake triggers
Image triggers
Sound triggers
Credits triggers
Preparing for standalone



Image
Displays an image when the camera it's connected to is active. Use the **Trigger Time** parameter to add a delay before the image is shown, and the **Image Duration** to specify how long the image should be displayed.

e.g. a **Trigger Time** of 1 and an **Image Duration** of 4 will delay the image showing for 1 second and then show it for 4 seconds.



Sequence
Displays a sequence of images one after another when triggered. To select an image sequence, use the ... button to navigate to and select the first image in your sequence. The frames must be numbered sequentially with no additional zeros prefixing lower numbers (e.g. 1, 10, 100 rather than 0001, 0010, 0100).

The **Trigger Time** parameter functions the same as with an image, delaying the displaying of the images.

The **Image Duration** parameter sets how long the image sequence should play for. The image sequence will loop if it's shorter than the **Image Duration**. The **Frame Time** defines how long each individual frame should display for. For a 25fps image sequence, each frame would be displayed for 50ms.

If you only want to play an image sequence once, work out how long the **Image Duration** will be for one complete run-through by multiplying the **Frame Time** by the total number of frames in the image sequence. For example:

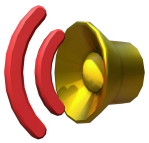
Number of frames = 125
Frame Time = 50ms per frame
125 x 50 = 6250ms for one run-through
∴ **Image Duration** needed = 6250

Sound triggers

▶ **Video Tutorial:** https://youtu.be/jlS7rp-7U-o

Sound triggers are a simple way to play or loop sound effects when a particular camera is active. They play whatever sound you have entered into the sound0 slot (found under the *Developer Settings* panel).



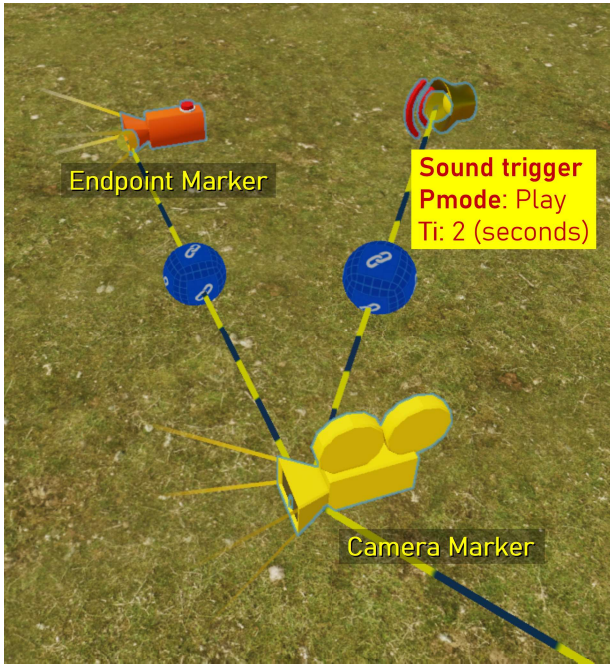


Sound trigger
Connect a sound trigger to the camera you want to play a sound during. Choose whether to play or loop your sound using the **Pmode** parameter. You can delay the start of playing with the **Ti** parameter.

Pmode | Either “play” or “loop”

Ti | The delay before triggering, e.g. 5.000 = 5 seconds.

Sound0 | Specify the sound to play or loop here.



Sound formats

At the time of writing, GameGuru MAX only supports two sound formats; 16-bit PCM uncompressed WAV files (.wav), and Ogg Vorbis (.ogg) files.

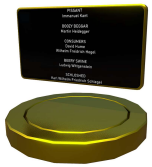
WAV files offer superior quality at the expensive of large filesize. Mono WAV files are the only sound format GameGuru MAX can use as '3D sounds', meaning their volume and left-right balance is dynamically adjusted according to the position of the camera. In the example, left, our camera moves past a sound trigger. Because we've specified a mono WAV file in slot 0, as the camera moves past, the sound will pan from left to right.

OGG files benefit from clever compression that keeps the filesize small whilst maintaining most of the quality. Mono OGG files are not supported by GameGuru's 3D sound system but will still play. We recommend using OGG wherever possible and only use WAV when you want a 3D sound.

Credits triggers

▶ **Video Tutorial:** https://youtu.be/CRebhe1ok3M

Use credits triggers to create a rolling credits effect for the end of your game. You can control all of the parameters from within the editor.



Credits trigger
Credits triggers work by displaying the contents of a text file on the screen, and moving the text up the screen. To use one, simply connect your credits trigger to a cine-trigger zone or cine-trigger entity and be sure to have a text file ready containing your credits.



Text File | Here you specify the text file to use as your credits.

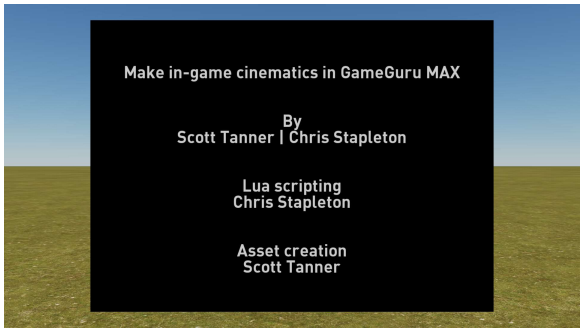
W | Rolling credits can have a black background. Here you set the width, in screen percentage terms, of that background.

H | Sets the height of the black background. It also defines how far the text scrolls. At 100, the text will spawn at the very bottom of the screen and scroll up to the very top. At 50, the text will spawn 25% of the way up the screen, and scroll up until it reaches 75% of the screen height, then disappear.

Sz | Defines the font size according to GameGuru MAX's lua text commands (1-9)

Spd | How quickly the text scrolls up, 1 being slow, 100 being very fast.

Here are some examples of how tweaking these parameters can yield different results:



W: 70 **H:** 90 **Sz:** 5 **Spd:** 20

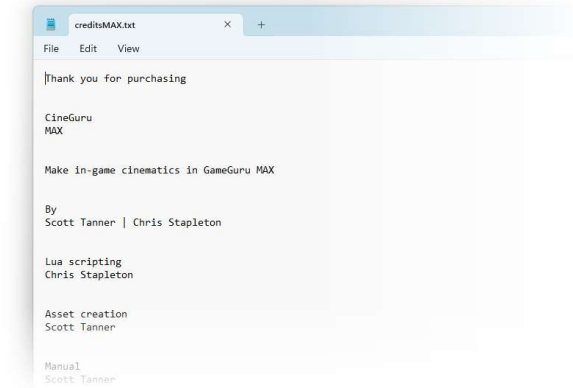


W: 0 **H:** 110 **Sz:** 5 **Spd:** 30



W: 100 **H:** 110 **Sz:** 3 **Spd:** 30

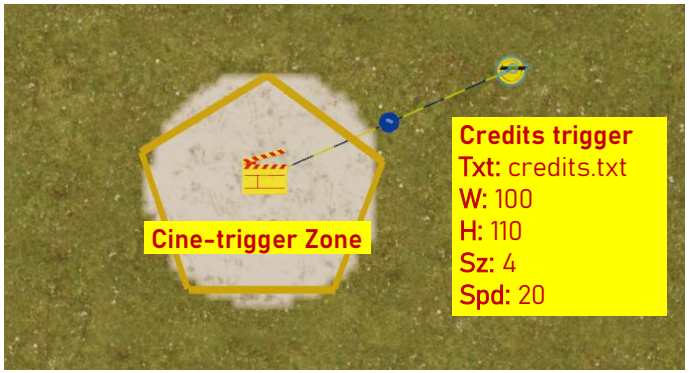
To set up rolling credits for your game, first create a text file with your credits in – we'll call ours *credits.txt*. Rolling credits will always be centred so you can't have multiple columns. The example, right, shows how you could write your credits.



In GameGuru MAX, add a cine-trigger zone or cine-trigger entity to your map, and a credits trigger. Connect the two using the Visual Logic Connections.

After we've selected credits.txt as our text file, we need to set a width and a height. We're going to set the width to 0 so there will be no background.

The height governs both the background height and the text scroll height. We want the credits to scroll the full height of the screen. We could type 100, but you'll notice the text appears to pop in at the bottom of the screen and disappear as soon as it reaches the very top. Instead, try a higher value like 110. This effectively spawns the text below the bottom of the screen and allows it to scroll up revealing it gradually – much more professional!



The text size and speed can be set to your taste. Here, we'll use size 4 and a speed of 20.

The result is pictured bottom right: when the player enters the zone, our credits text will gradually scroll up the screen. Note that, at the moment, this text appears over the player camera, and the player can still roam about the level freely.



As an optional final step, you can drop in a camera marker and connect it to the same cine-trigger zone as our credits trigger. When activated, the credits will play over whatever the camera is pointed at. If your camera marker is the first of a series of shots, the credits will continue to display on top of all of them.


PREPARING FOR
STANDALONE

Fpp files

▶ **Video Tutorial:** https://youtu.be/fmg01-cPx0g

GameGuru MAX remains in constant development, so it's advisable to export standalones of your levels often, to check what is and isn't working. If your level contains any **Cine|Guru MAX** cinematics, be aware that certain files will not be included by default in a standalone build. These include:

- Subtitles text files
- Music files specified only in action texts (e.g. those control by the music, mplay, mvol, and mstop commands)
- Image sequences (although single images should be fine)

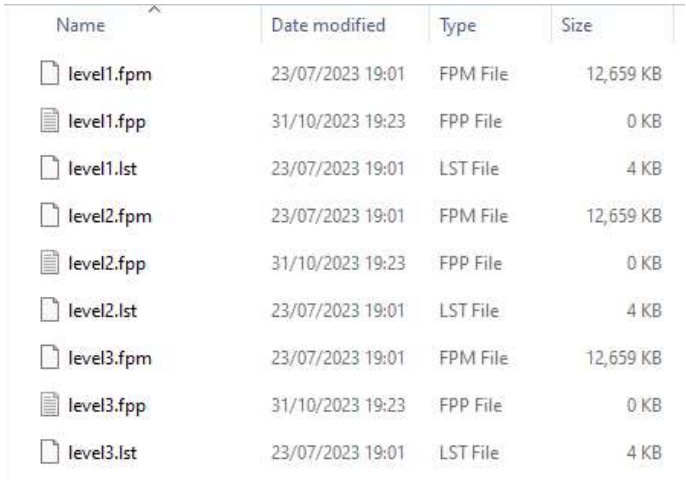
If your standalone doesn't include these files, you may experience errors when playing your game, or it may even crash it entirely. You could transfer the missing files over manually, but is time-consuming and the copied files will not be encrypted. Fortunately, there is a way you can force GameGuru MAX to include certain files within a standalone build: the FPP file!

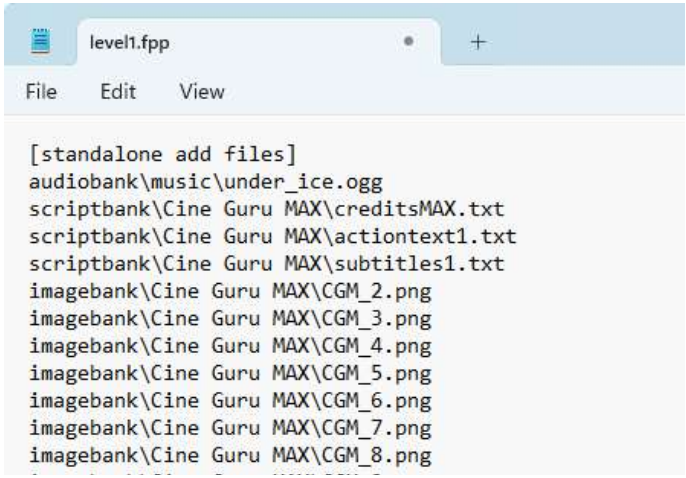
The FPP file is essentially a text file that sits alongside your FPM map file in mapbank, which contains a list of files to include (and not include) whenever that map is added to a standalone.

Creating an FPP file
To create an FPP file, simply make a new text file and rename it to the name of your level, with the file extension .fpp. If you have multiple levels, you may wish to include one FPP file per level, or you can include it all in the first level's FPP file.

Open your newly-created FPP file and type [standalone add files].
Then on a new line, give the exact filepath and filename to every file you wish to include. For image sequences, this means specifying the filepath and filename of each image in the sequence one by one.

Since you specify action texts in the editor, they will be included in standalone builds, but files referenced only within action texts, such as subtitle text files, and music files, won't be included so remember to specify those in your fpp file.





After you've listed every file you wish to add, you can also write the command [standalone delete files] to omit certain files from the standalone – although it's less likely you'll need this.

Lastly, it's worth mentioning that text files are not and can not be encrypted by GameGuru MAX; this is because the various lua scripts that rely on them cannot handle encrypted files, unless the lua script itself has GameGuru MAX's encryption key. Making this public would compromise everyone's GameGuru MAX games, so we've elected not to push for this feature to be added.

These files are not supposed to be used to store any critical or personal information, but it's just something to be aware of before you click “Build Standalone”.

Cinematics as videos

Cine|Guru **MAX** has been designed to allow users to create live in-game cinematics. However, some users may wish to use these tools to create sequences which can be recorded with a screen recorder, and played back using a standard GameGuru MAX story zone. This approach is known as “pre-rendering” and has some advantages, but also some disadvantages.

Pros

- Pre-rendered cinematics always look the same regardless of the end user's hardware.
- You can have more control over sound effects and graphics with a pre-rendered cinematic.
- You can more easily edit and rearrange the shots in your sequence, even add additional footage not shot in GameGuru MAX.

Cons

- GameGuru MAX stretches video to the end user's screen resolution. If you make a cinematic in 16:9 and someone plays your game on a 21:9 monitor, it will be stretched horizontally.
- Pre-rendered cinematics cost more video memory than an in-game cinematic.
- Running screen recording software at the same time as GameGuru MAX can lower your framerate, causing actions to go out-of-sync during your recording.

GameGuru MAX is more forgiving than GameGuru Classic when it comes to supported video formats. We recommend you use .mp4, but you're now no longer limited to only 25fps videos; the common framerates of 24, 25, 30 and 59.94 fps are all now supported.

Regardless of whether you adopt the in-game or pre-rendered approach, we hope you have fun creating your own in-game cinematics and scripted sequences using **Cine**|Guru **MAX**!



Scott Tanner | Chris Stapleton

Official support Discord
https://discord.gg/mVgdh49xxP

Copyright © 2024 Scott Tanner and Chris Stapleton