1. What is Actimator?

Actimator is a Web-based, collaborative, visual game development software for making and publishing 2D games. The primary focus of Actimator is being educational — easy to learn, in particular for those with no coding skills — without compromising the quality of games that can be built.

Using Actimator’s visual environment, you can:

• Practice video game design and programming skills.
• Create a computer game with your friends.
• Code together and learn coding from each other.
• Add physics such as gravity, density, and forces to your game.
• Publish your game on the Actimator Website and share it with your social network.
• Playtest your game on iPhone, iPad, and Android devices, using Actimator Player app.
• Export your game for iOS App Store and Google Play Store.

In This Chapter

1.1 Goals and Features

Actimator is made with the following goals in mind:

**Easy to Learn:** Learning how to make a video game should not be hard. Anyone over the age of 10 should be able to learn how to create and publish a video game.

**Fun to Make Games:** Making video games should be as fun as playing video games. Our motto is: *don’t just play games, make one.*

**Get Results Fast:** Making video games should be more about turning your creativity into a playable game, and less about struggling with the underlying game and physics engines.

**Be Collaborative:** Making games requires multiple skills such as storytelling, level design, programming, graphics design, and sounds design, that are rarely found in a single person. A good game design software should help a team to work closely together.

We have built the following features into Actimator to achieve these goals, without reinventing the wheel.

1.1.1 Educational

Actimator is specifically optimized for beginners with no coding experience. This book and tutorials are designed to be self-paced for absolute beginners. We follow the *Learning by Doing* mantra to teach the required coding skills to beginners, as they use Actimator.
Actimator Book of Game Design,

1.1.2 Web-based Environment

Actimator is built completely using HTML5 technologies and runs inside a Web browser. You do not need to install any plugin in order to run Actimator. Besides the game development environment, Actimator Website takes care of organizing and storing your projects, sharing your projects with your friends, and publishing your projects to the Web and mobile platforms.

1.1.3 Rapid Development Environment

Actimator’s visual environment makes it fast and easy to design and develop video games. The visual environment includes visual scene (level) editor, visual physics editor, and visual programming language.

Being completely visual, Actimator lets you make games as complex as Angry Birds in hours or days, rather than weeks or months. Furthermore, being real-time collaborative, it lets you and your team work together on the same project at the same time, hence increasing your productivity and decreasing development time.

1.1.4 Built-in Physics

Actimator comes with a built-in physics engine built on top of Box2D, which enables developing complex physics-based games such as Angry Birds. Furthermore, Actimator’s visual physics editor makes it easy to define and adjust physics behaviors such as gravity, collision, speed, bounciness, etc.

1.1.5 Realtime Collaboration

Actimator is realtime collaborative. This enables multiple users to work on the same game project at the same time, no matter where they are located. It works quite similar to Google Docs; only here users will be working on a game project. When working on a game together, changes made by a user are instantly applied to all other users in a fraction of a second. This allows users to code together, see each other’s changes immediately, learn from each other’s actions, or divide the game design and coding tasks between each other.

Real-time collaboration brings the following advantages to learners and teachers:

- **Social Engagement**: Which one is more enjoyable: working on a project alone all day long, or working together with your friends towards the same goal?
- **Peer Learning**: You can learn coding skills from a more advanced peer, by watching the code unfolds in front of your eyes as he/she writes it. Or vice versa, you can teach coding to a friends, by walking them through a tutorial and helping them in coding/design activities.
- **Pair Programming**: It has been shown that pair programming leads to higher quality products. Working in a pair or a team increases the chance of producing a game that is actually playable.
- **Productivity**: Which one is faster and/or results in a better game: one person doing all the design, coding, and playtesting tasks, or two friends working on the same project, either doing the tasks together or dividing the tasks between each other?

1.1.6 Cross-Platform Publishing

Actimator offers Web, iOS, and Android publishing. You can save and maintain multiple versions of a game and publish a new version when it’s ready.

- Each game gets a page with a clean URL that you can use to showcase and share your games with your social network.
- You can playtest your game on iPhone, iPad, and Android devices, using Actimator Player app.

1.2 Structure of This Book

This book is meant to be self-directed to teach you how to use Actimator. It does not assume any prior knowledge in programming or game development. As you go through the book and tutorials, you will gain skills that you need to create fairly sophisticated games.

Furthermore, this book is intended to serve as an introductory course in computer science and programming. The programming concepts used in Actimator’s programming language and this book are mostly universal. Therefore, by
learning Actimator, you will gain coding skills that you can apply in the programming languages that you might use in the future, such as JavaScript and Python.

The rest of this book consists of the following chapters:

• **Getting Started:** learn how to make your first project.
• **Actimator’s Game Design Elements:** get yourself familiar with Actimator user interface, tools, and features.
• **Tutorials:** include steps for creating simple yet complete games.
• **Actimator Programming Language:** details of Actimator’s programming language, it’s game engine, and physics programming.
• **Advanced Programming Topics:** contains advanced topics and knowledge that are not required when you first start using Actimator.
• **Publish Your Game:** walks you through publishing your game on the Web and mobile platforms.
2. Getting Started

2.1 Signup

To start creating games, you must create an account on the Website. You can sign up with your email address or with Facebook.

Your birth year is required for sign up. If you are younger than 13 years old, you will have to provide your guardian’s email address.

2.1.1 Login

Once signed up, you can sign in using your username/password.

2.2 Create a Project

Once signed in, you can go to My Projects page to see the list of projects that you have created or have been shared with you.
From the top of ‘My Projects’ page, click on the Create a New Project button.

Enter the project information, including a domain name. Your game will have a page at /app/[domain]. This makes it easy to share your game page with others. Moreover, when you publish your game on the Web, your game will be available on this page for others to play.

The project name, description, and image appear on the project page.

The project splash image appear when the game is being loaded.

Press the Create button to create the project.

2.3 A Tour of the Lab

Once you create a new project, it appears in the “My Projects” page. To work on your game, click on the Lab link at the bottom of the project. This will take you to the game design environment (the Lab).
All development for your game happens in the Lab. The Lab contains three main areas; the Top Menu, Bottom Menu, and the Gallery.

### 2.3.1 The Gallery

The Lab contains the Gallery, which is divided into Assets, Actors, and Scenes tabs.
In the Assets tab, you can upload and preview the image and audio assets for your game. The assets you upload define how your game looks and sounds.
In the **Actors** tab, you can create actors and edit their properties, you can remove actors, and you can code an actor’s program.
In the **Scenes** tab, you can create and remove scenes, you can set the starting scene for your game, and you can open a scene in the **Scene Editor**.
2.3.2 The Programming Area

In the Programming Area, you can code how an actor behaves.

2.3.3 Collaboration

The Attendants icon at the bottom-right of the screen allows you to view how many users are in the session. When you click on it you can see a list the users with their names and avatars.

The Chat icon located at the bottom-right of the screen. When clicked, it opens the chat box.
The avatar of the user making a change (in the Scene Editor or the Programming Area) is displayed above that change for all users to see.

2.3.4 Top Menu

Invite:

You can invite others to join your project. Just press the invite button and enter the email addresses of everyone you want to invite. Each person will receive an email indicating that you have invited them as a collaborator for this project. Once they accept the invitation, they can access the project and join you as a collaborator from their “My Projects” page.

Invite by Email:

bob@yahoo.com alice@gmail.com

Save, Autosave, and Versioning:

Actimator autosaves frequently. If you wish to save at specific time, just click the Save button.
To save a version of your game, click the down arrow on the save button and input a version number and description of changes you made in that version.

Notifications:
The notification bar at the top of the screen display the most recent change made to the project. When clicked, it displays a list of all the changes made during the session.

Publishing:
Once satisfied with your game, you can publish it:

You will have the choice to publish to the web (HTML5), to the App Store (iOS), and to the Play Store (Android).

Closing a Project:
To return to your projects page, press the close button.

2.3.5 Bottom Menu

The menu on the bottom-left the screen has four options.

- Volume – Controls the volume of Actimator
- Bugs – Allows you to submit a bug report
- Quick Link – Provides a URL to the current project your working on
- Settings – Allows you to set the device’s screen orientation
3. Actimator’s Game Design Elements

3.1 Overview

This section provides a quick overview of all the Actimator’s game design elements.

- **Assets** are images and audios used in a game. Asset files must be uploaded into the project to be used in the project. Learn more about Assets.
- **Actors** are game characters. An Actor consists of an image, physics and image properties, and its program. Learn more about Actors.
- **Scenes**: A game is made out of one or more scenes. You can think of each scene as one level of the game. Each scene contains background/foreground images, background audio, and entities. Learn more about Scenes.
- **Entities** represent game characters. Each entity is a copy of an Actor. We can draw an actor multiple times on the scene, each of which is an entity. Learn more about Entities.
- **Program**: Each entity has a program that defines the behavior of that entity during the game. Entities use the program of their Actor for their behavior. This means entities that are created from the same actor will behave the same. Learn more about actor’s program.
- **Physics**: Each entity has its own physics properties such as shape, density, bounciness, speed, and rotation speed. This allows you to easily and quickly create games similar to Flappy Bird and Angry Birds. Learn more about Physics.

3.2 Assets

Assets, simply put, are images and audio. Each asset is an image or audio file that can be uploaded from your computer into your project and used in your game.

3.2.1 Images

Image assets are used for backgrounds or foregrounds of a game scene, depicting game characters, and making animations.

**Acceptable image file formats**: Actimator accepts PNG, GIF, JPEG, and BMP files. For game characters (actors), it is recommended to use PNG files as PNG is the only format that allows images with a transparent background.

3.2.2 Audio

Audio assets are used for scene background audio, or sound effects.
**Background Audio:** For each scene, we can choose an audio file to be played as soon as the scene is started and as long as that scene is running. This is called a background audio. When an audio file is selected as background audio, it automatically restarts as soon as it ends.

**Sound Effects:** Sound effects (Sound FX) are played on certain occasions in the game, such as when the player touches an actor or when two actors hit each other. Sound effects are played using Actimator’s Audio commands from within an Actor’s program.

**Acceptable audio file formats:** Actimator supports MP3, WAV, and OGG file formats.

### 3.2.3 Uploading Assets

Each asset is an image or audio file that can be uploaded from your computer or a Google Image Search. To upload an asset file:

- Click on the Assets tab.
- Press the button.
- Select files to upload to your project.

**Multiple File Upload:** Actimator makes it very easy to upload a lot of assets quickly into a project. To do this, you can select all files at once and upload them together. Actimator takes care of uploading all selected asset files. You can even upload images and audios together. When the uploading process is complete, all uploaded assets appear under images and audio sections.

**Google Image Search:** Actimator helps you quickly find image assets on Google Images. Just enter the name of the asset you are looking for and press the search button. Actimator uses Google Image Search to search for images that are "Labeled for reuse with modification". When you find an image you like, save it to your desktop and upload it to your project. Make sure you understand the terms and have permission to use that image in your project.

### 3.2.4 Previewing an Asset

You can preview each asset. For images, move your mouse on the ? icon next to each image. For audio, press the play icon next to each audio.

### 3.2.5 Replacing an Asset

You can replace a single asset by pressing the replace button next to that asset and selecting a new file. After the new file is uploaded, the current asset will be replaced by the new asset.

### 3.2.6 Renaming an Asset

To rename an Asset:

- Double-click on the Asset’s name.
- Enter the new name for the asset.
- Press Enter key.

### 3.2.7 Removing an Asset

To remove an asset, press the x button next to the asset.

### 3.2.8 Asset Optimization

- **Quality:** Actimator automatically resize the scene to fit/fill in the device screen. This can impact the quality of the game on larger displays if uploaded images are in low resolution. Make sure that uploaded images have enough quality to look perfect on large tablets and desktops.
- **File Size:** The size of asset files can highly impact the loading time of your game. Make sure to optimize asset files before uploading them to Actimator.

Adding Background Images
3.3 Actors

Actors are game characters. An actor consists of an image, physics and image properties, and its program.

Actors are different from images in that images do not interact with any other image or actor on the scene. They do not move or change, do not have physics properties, and do not have a program. But actors can move, hit other actors and bounce back, play a sound, change their image, and so on.

A defining feature of actors is their program. With their programs, actors allow us to create the interactions in games. An actor’s program is created using Actimator’s Visual Programming Language.

3.3.1 Creating Actors

To create a new actor:

- Click on the Actors tab
- Press the Add Actor button

A New Actor window appears, here we can set all of the actor’s properties.

- Name the actor
- Select an image for the actor
- Modify actor’s properties if needed
- Press the Create Actor button

![Image of creating a new actor in Actimator]
3.3.2 Modifying Actor’s Properties

We can always modify actor’s properties by clicking on the edit icon next to the Actor. Actor’s properties are not directly used in the game. However, when an Entity is created, either when making the game, or when the game is running, it automatically copies its actor’s properties.

Design time: When creating multiple entities of the same Actor on the scene, you can save time in setting their properties individually, by setting the Actor’s properties correctly.

Run time: When an entity is created using the program, it copies all initial values for its properties from its Actor properties. Therefore, by setting actor’s properties properly, you can make sure that any entity that is created during the game will appear and behave on the scene as you expect.

3.3.3 Renaming an Actor

To rename an actor:
- Double-click on the actor’s name.
- Enter the new name for the actor.
- Press the Enter key.

3.3.4 Removing an Actor

To remove an actor, press the x button next to the actor.

3.3.5 Accessing Actor’s Program

Each actor has its own program. To access an Actor’s program, press the program icon next to that actor. Learn more about actor’s program.

See also:
- Modifying Entity’s Properties

3.4 Program (code)

A program, also known as code, is a way to define how actors behave while a game is running. An actor’s program consists of events, commands, and control flow.

3.4.1 Events

Actimator’s Visual Programming Language relies heavily on events. Actors wait until one of the events in their program is triggered. Once an event is triggered, the commands contained inside that event are executed.

3.4.2 Commands

Commands control how actors and scenes behave while the game is running and are executed, one at a time, in order. The commands in a program can control how entities look and move, how the game switches between scenes, etc.

3.4.3 Control Flow

Control flow allows commands in an actor’s program to be executed when certain conditions are met.
3.5 **Scenes**

Scenes are where everything in your game comes together. They can be menus, levels, or you can make a whole game in one scene. You can drag images, actors, and sounds to the scene. Scenes are split into layers that control the order in which images are laid on the scene. Each scene has its own properties that can be set like; Size, Scale, Aspect Ratio, and Gravity.

3.5.1 **Creating a Scene**

To create a new scene:
- Click on the Scenes tab
- Press the **Add Scene** button
- Give the scene a name
- Press the **Create** button

Starting scene:
When you hover your cursor above a scene a flag appears. When you click the flag next to to a scene, that scene is set as the starting scene.

3.5.2 **Editing a Scene**

To edit a scene, just click on the scene’s name.

3.5.3 **Removing a scene**

To remove a scene, press the × button next to that scene.

3.5.4 **Renaming a scene**

To rename a scene:
- Double-click on the scene’s name.
- Enter the new name for the scene.
- Press the **Enter** key.

3.6 **Entities**

What is an entity?

Entities are what make your game interactive. Each entity is a unique copy of an actor: it inherits all of the actor’s properties and program.

3.6.1 **Creating Entities**

At design time: When you drag an actor onto a scene, an entity of that actor is created. If you wish to create several entities quickly, you can select the **Draw Tool**.
- Click the **Draw** button.
- Select an actor.
- Each time you click on the scene you will create an entity of the selected actor.

At run time: In many situations, we want entities to be created during the game. This can be achieved using the **Entity Management commands**. An entity can be programmed to create other entities. It can even set position, size, and other properties of the entities that it creates.

3.6.2 **Copying Entities**

You can copy an entity by holding down the Ctrl key and dragging a copy of it away from the original. The copied entity will inherit all of the properties from the entity you copy it from instead of its actor.
3.6.3 Entity's properties

Each entity has its own properties such as its size, image, position, or shape. When an entity is created, it copies all the properties of its actor. But then those properties can be modified.

You can see these properties by clicking on an entity. An entity’s properties appear in the property editor.

3.6.4 Modifying Entity’s Properties

Design time:

You can easily move and resize entities on the scene. If you want to be more precise, you can set exact values in the Property Editor, you can also set all other an entity properties.

Run time:

Entities can change while the game is running. In an actor’s program, you can specify which properties of which entity you want to change.

A few notes:

• Changing properties of an entity, either on the scene or through the program, does not affect properties of other entities, nor the properties of its actor.
• Changing properties of an Actor does not affect the properties of existing entities on the scene. But any entities of that actor that are created from now on, will copy the new values for those properties. This includes both new entities created on the scene and through the program.

When a new entity is created using the program, and you need to change its properties to have a different value from its initial value (that is copied from its Actor’s properties), you can use the Create event to achieve that.

3.6.5 Removing an Entity

To remove an entity from the scene, click on the entity and press the Delete key. Use the Erase tool if you want to remove several entities quickly. After clicking the Erase button, every entity you click on will be removed.

3.7 Physics

3.7.1 Overview

Actimator comes with a built-in physics engine, which allows creating sophisticated games similar to Angry Birds in an easy and quick way.

Physics Scale (pixels per meter)

In the scene’s properties, you can set the Scale (pixels/meter) parameter. By default physics scale is set to 30, this causes the physics engine to simulate 30 pixels as on meter. For example, If and entity is 60 pixels wide, the physics engine will perform calculations for that entity as if it were two meters wide.

3.7.2 Enabling/Disabling Entity’s Physics

Each entity has a “Has Physics” property that, when set, causes the entity to participate in physics simulation. If an entity has physics enabled, it can move or rotate with a given velocity, collide with other physics entities, fall from the scene’s gravity, etc. If an entity does not have physics, it won’t react to other entities or physics forces.

You can enable/disable an entity’s physics at any time, there are a few ways to do this:

Actor Properties:

You can enable/disable physics in an actor’s properties. All entities created from an actor inherit that actors physics setting, entities that are already created retain their own physics settings.

Entity Properties:
If you have an entity created on the scene already, you can change its physics setting in the Property Editor. Only that entity’s physics will be changed.

Actor’s Program:

You can enable/disable physics in an actor’s program when a certain event is triggered. All entities of that actor will change physics settings when their programmed event is triggered.

3.7.3 Collision

What is a collision?

In Actimator, a collision occurs when the collision boundary of one entity overlaps another.

What happens during/after collision?

Collision start:

When two entities first collide, the **ON_STARTCONTACT** event is triggered. Any code contained in the **ON_STARTCONTACT** event will be executed at this time. This is also a good time to play collision sound effects.

During collision:

The **ON_IMPACT** event is triggered for the duration of a collision and is mostly used for checking the force of a collision. The physics engine will automatically try to move the entities apart until they are no longer colliding. The faster the physics engine has to move entities away from each other, the stronger the force of the collision. This force can be returned from the **Impact Normal Impulse** parameter.

After collision:

The moment colliding entities are no longer overlapping, the **ON_FINISHCONTACT** event will be triggered.

3.7.4 Adjusting Collision Boundaries

Regardless of how an entity looks, its collision boundary determines how it behaves in physics simulation. In the scene editor, collision boundaries show up as either a blue circle or square outlining each entity. There are options to set the size and shape of the collision boundary.

Physics Shape:

You can set the physics **Shape** of an entity to a “Square” or “Circle”.

Collision Width and Height Scale:

You can adjust the collision scale of an actor to resize the collision boundary to better match its image. **Collision Width Scale** and **Collision Height Scale** can be set to any number above 0. For example “1” equals 100%, “0.5” would be 50%, and “2” would be 200%.

Image Offset:

To better match up an actor’s image with its collision boundary, you use **Image Left Offset** and **Image Top Offset**. These values can be a positive or negative number that offsets the image of actor from its collision boundary.

3.7.5 Physics Type

There are two options you can set for physics type, “Static” and “Dynamic”.

Static Entities:

Entities that have their physics **Type** set to “Static”, are not able to move and are not affected by gravity. Other physics entities can collide with, and will bounce off of a static entity.

Dynamic Entities:

Entities with a “Dynamic” **Type** are affected by all physics forces and all other physics entities.
3.7.6 Making Entities Move with Physics

Linear velocity (Speed)

The **Speed** parameter sets the speed of an entity in meters per second.

Velocity direction (Speed Direction)

You can specify the direction to apply the speed force to an entity with the **Speed Direction** parameter, this can be a value between 0 and 360 degrees.

3.7.7 Applying Gravity to the Scene

You can apply gravity to a scene with the **Gravity -X** and **Gravity -Y** parameters, this will apply gravity to all dynamic entities. The value for the gravity parameters is set to meters per second per second.

3.7.8 Making Entities Move Without Physics

Although “Static” entities and non physics entities cannot move, you can still update their positions rapidly to give the illusion that they are moving. Just increase/decrease the x or y position of an entity on every tick or render event.

3.8 Scenes - Advanced

3.8.1 Property Editor

The Property Explorer has three tabs: Scene, Entity, and Image. You can click on the scene tab at any time to set scene properties. When you click an entity or image on the scene, its specific properties appear in the Property Explorer.
### 3.8.2 Tools

- **Select** – Allows you to select entities and images on the scene
- **Camera** – Controls the viewport’s position on the scene
- **Draw** – Draws entities on the scene with one click, instead of dragging them
- **Erase** – Erases entities on the scene with one click
- **Stop Scene** – Stops the scene from running
- **Play Scene** – Plays or pauses the scene
- **Rotation** – Sets the viewport to either portrait or landscape
- **Navigation** – Opens and closes the navigation window
- **Background Image** – Adds or changes the background image
- **Background Audio** – Adds or changes the background audio

### 3.8.3 Dimensions

**Width and Height:**

You can resize your scene by setting the width and height. Keep the game orientation (portrait or landscape) in mind when sizing the screen. Depending on the size of your scene, it may not completely visible. The default
size of a scene is 1280 x 720.

Aspect Ratio:

Aspect ratio is the ratio between the width and height of an image. The default aspect ratio in Actimator is 16:9, which is the same ratio as the default width and height. Many devices have screens with different aspect ratios.

3.8.4 Viewport

The viewport is the visible portion of your game. You can adjust the viewport to see how your game might look on different devices, you can also adjust how the scene is scaled to the viewport.

You can set the Scaling Method of a viewport in the scene’s properties:

- “Fit in Viewport” - This option will make the scene fit inside the view port while still preserving the aspect ratio.
- “Stretch to Viewport” - This option will stretch the scene to fill the entire viewport, this can cause images to be distorted in there’s a big difference between the scene’s and the viewport’s aspect ratio.
- “No Scaling” - This option will keep the scene’s size constant, even if it is smaller or larger than the viewport. If the scene is larger than the viewport, only the first 1280 x 720 pixels will be visible. You can view more of the scene by using the camera parameters.

Padding:

Padding is the area around your scene if it doesn’t completely fill the viewport. You can change the padding color in the scene properties editor.

3.8.5 Navigation

The navigation window shows a small preview of the entire scene and outlines what part of the scene is visible in the viewport with a white rectangle.

3.8.6 Camera

If your scene is larger than the viewport, you can reposition the viewport on the scene using the Camera X and Camera Y parameters.
Tutorials are replaced with Actimator Courses.
5. Actimator Programming Language

5.1 Events

Topics:
- What is Event-Driven Programming?
- Actimator Events
- Actimator's Game Engine Execution Cycle
- Description of each event
- Custom Events
- Event Command

5.1.1 What is Event-Driven Programming?

Event-driven programming is a programming paradigm that executes commands when certain events are triggered. Events can be triggered by human input or even from within the program itself.

5.1.2 Actimator Event Types

Actimator’s visual language uses Event-driven programming. The visual language consists of commands that control everything from how entities and scenes look or move. These commands are executed when certain events are triggered. Here are some of the events:

- User Input (Keyboard, Mouse/Touch)
- Entity (Create/Destroy)
- Physics (Collisions)
- Graphics (Before/After Render: 60 times per second)
- Timer (Tick: 20 times per second)
- Device (Rotation, Resize Screen)
- Custom (user-defined)

5.1.3 Actimator's Game Engine Execution Cycle

Actimator’s game engine is the framework that allows you, the developer, to easily create games. The engine consists of an HTML5 canvas renderer, Box2D physics engine, and Actimator’s visual programming language. The engine cycles through iterations 60 times per second.

At the beginning of each iteration is the Execution Step:

- The Execution Step begins by calculating physics, for example, if an entity is colliding with another or a movement is being performed. During this physics calculation, Contact Events and the Impact Event can be triggered.
- The Before Render Event is triggered prior to the render.
- The physics calculation is followed by render-
Actimator Book of Game Design,
ing. During the render, images are drawn on
layers. If a layer is set to be updated, it will be
cleared and then redrawn.
• The After Render Event is triggered when the
render completes.

Following the Execution Step, non-Step events can be
triggered in the remaining time of the iteration. These
events include, in no particular order:
• User Input – If the player happens to click with
the mouse, presses a key on the keyboard, or
touches the screen of the device at any time dur-
during an iteration
• Tick Events – The Tick Event “ticks” 20 times
per second independent of game engine’s cycle.
If one of these ticks occurs during an iteration,
the Tick Event is triggered
• Custom Events – As the developer, you can cre-
ate custom events. If a Run Command is exe-
cuted in any of these events during the iteration,
a custom event corresponding to that Run Com-
mmand is triggered.

5.1.4 Description of each event

On Create
This event is triggered once when an entity is created.

On Destroy
The On Destroy event is triggered once in the instant
before the entity is destroyed.

On Tick
The Tick Count is increased by one, 20 times per sec-
don, from when the game starts. The Tick event is
triggered each time the Tick Count is increased.

On Before Render
This event is triggered right before graphics are ren-
dered on the screen(60 time per second).

On After Render
This event is triggered right after graphics are rendered
on the screen(60 time per second).

On Start Contact
This event is triggered once the instant that two entities’
collision boundaries overlap each other.

On Finish Contact
This event is triggered once the instant that two enti-
ties’ collision boundaries are no longer overlapping
each other.

On Key Down
This event is triggered once when any key is pressed.
You can use an If Statement to specify which key is
being pressed.

On Key Up
This event is triggered once when any key is released.
You can use an If Statement to specify which key is
being released.

On Mouse Down / Touch Start
This event is triggered once when the mouse clicks on
an entity or when the player touches an entity on their
device’s screen.

On Mouse Up / Touch Release
This event is triggered once when the mouse click is
released on an entity or when the player releases an
entity on their device’s screen.

On Scene Mouse Down / Touch Start
This event is triggered once when the mouse clicks
anywhere on the scene or when the player touches
anywhere on their device’s screen.

On Scene Mouse Up / Touch Release
This event is triggered once when a mouse click is
released anywhere on the scene or when the player
releases anywhere on their device’s screen.

5.1.5 Custom Events

Custom Events are useful for simplifying redundant
tasks. There may be an entity in your game that you
want to jump, play a sound, and change its image when
its clicked on or a key is pressed. This could be ac-
complished by dragging the code into a custom event
called “Jump”. You could then use the Run Command
to run the “Jump” event when a mouse or keyboard
event is triggered. You can also trigger non custom
events using the run command.

Run CUSTOM_EVENT_1

In some situations, you may need to run an event from
another entity. You use the Run Entity’s Event Com-
mmand for this. Just type in the Name of the entity and
the event that you want it to run.

Run Entity IName CUSTOM_EVENT_1

Custom Events are created the same as all other events.
After creating a custom event, you can rename it by
double-clicking on its name. Type in a desired name
and press enter on the keyboard.

CUSTOM_EVENT_1

5.1.6 Event Command

The event command returns a value that is influenced
by an event.
You can use the drop-down menu to select which value you want:

**Tick Count**

The Tick Count increases by one, 20 times per second, from when the game starts.

**Mouse/Touch X**

Returns the X position on the scene of either the mouse or a touch.

**Mouse/Touch Y**

Returns the Y position on the scene of either the mouse or a touch.

**Collided Entity’s Id**

Used in one of the On Contact events, this returns the Name of the entity who’s colliding with This Entity.

**Collided Actor**

Used in one of the On Contact events, this returns the Actor who’s colliding with This Entity.

### 5.2 Assignment

**5.2.1 Set Command**

The Set Command, sets one value, variable, parameter, or mathematical expression to another.

Here’s some examples of how the Set Command can be used:

<table>
<thead>
<tr>
<th>Event</th>
<th>Tick Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Mouse/Touch X</td>
</tr>
</tbody>
</table>

### 5.2.2 Increment Command

You may need to set a value to itself plus one. Instead of using the Set Command, in this case, it’s more effective to use the Increment Command. The Increment Command takes a value (number, variable, parameter) and increases, decreases, multiplies, or divides it by another value (value, variable, parameter, mathematical expression).

In the example pictured below, the Increment Command increases the Score Variable by 100. The Set Command accomplishes the same result with more effort.

Keep in mind. Decreasing a value by a negative amount results in a positive number.

### 5.3 Logic

**5.3.1 When to Use Logic Commands**

Logic Commands can only be used within a Control Flow Statement. Logic Commands allow a Control Flow Statement to check if a condition is met or how values relate to one another.

**5.3.2 Relational Operator**

The Relational Command uses a relational operator to test the relationship between two numbers, variables, parameters, or mathematical expressions. This relation can return either true or false.
You can choose from these relational operators:

- == Equal to
- != Not equal
- > Greater than
- < Less than
- >= Greater than or equal to
- <= Less than or equal to

5.3.3 Logical Operators

Binary Operators

Logical Conjunction (AND)

The logical conjunction, AND, returns true only if both of its conditions are true. This can be represented with a truth table:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

Logical Disjunction (OR)

The logical disjunction, OR, returns true if one or both of its conditions are true. This can be represented with a truth table:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

Unary Operators

The NOT logical operator is used to check if a condition is not true. When a condition preceded by a NOT is false, it returns true.

The following checks to see if the My Name Variable is not equal to “Mike”. If this condition is true, the Score Variable will be set to -1.

5.4 Math

Topics:

- Mathematical Operations
- Unary Math Functions
- Binary Math Functions

5.4.1 Mathematical Operations

This Math Command takes two input values, these include; numbers, strings, variables, and parameters.

You can select a mathematical sign from the drop-down menu; “+” Addition, “-” Subtraction, “*” Multiplication, “/” Division, and “%” Modulo. The Math Command performs the desired arithmetic on the two input values and returns the sum, difference, product, quotient, or remainder of division between these values.

The following is an example of how you can use modulo. We check to see if Tick Count mod “%” 20 equals 1, which it will, once a second. We can then increase the Timer variable by 1 every second.

Modulo Operation:

The Modulo Operation, mod or % for short, returns the remainder of division between two values. For example 24 mod 9 is 6, because 24/9 = 2, 2*9 = 18, and the remainder is 24-18 = 6. This is very useful in programming. Another example would be if we wanted to find out if a number is odd or even. We can do any number mod 2. If the number is even, the result will be zero, otherwise it will be one.

More information on modulo can be found here: Modular Arithmetic Wikipedia
5.4.2 Unary Math Functions

Unary Math Functions take only one input value (number, variable, parameter), and apply a function selected from the drop-down menu:

- **ABS** – Returns the Absolute Value of the input value, this will always be a positive number
- **FLOOR** – Returns the input value Rounded Down to the nearest whole number
- **CEIL** – Returns the input value Rounded Up to the nearest whole number
- **ROUND** – Returns the input value Rounded to the nearest whole number
- **SQRT** – Returns the Square Root of the input value
- **LN** – Returns the Natural Logarithm of the input value with $e$ as the base
- **SIN** – Returns the Sine of the input value
- **COS** – Returns the Cosine of the input value
- **Tan** – Returns the Tangent of the input value
- **ASIN** – Returns the Arc-Sine of the input value
- **ACOS** – Returns the Arc-Cosine of the input value
- **ATAN** – Returns the Arc-Tangent of the input value

5.4.3 Binary Math Functions

Binary Math Functions take two input values (numbers, variables, parameters), and apply a function selected from the drop-down menu:

- **Random** – Returns a Random number between the two input values
- **Min** – Returns the Minimum or lowest number of the two input values
- **Max** – Returns the Maximum or highest number of the two input values
- **Power** – Returns the first input value to the Power of the second input value
- **Log** – Returns the Logarithm of the first input value with the second input value as the base
- **Atan2** – Returns the Arc-Tangent using two input values

5.5 Variables

**Topics:**

- What is a Variable?
- Global Variables
- Local Variables

5.5.1 What is a Variable?

In Actimator, a variable represents data (numbers, strings, booleans). Each variable is associated with a name, these names are used to reference the data contained within the variable.

5.5.2 Global Variables

Press the plus button on the Global Variable Command to create a new variable.

Type in the desired name for the variable:

Once set, the value contained in a Global Variable can be accessed by any other entity in the game.

5.5.3 Local Variables

Press the plus button on the Local Variable Command to create a new variable.

Type in the desired name for the variable:
Local Variables can only be accessed by the entity they’re created in. The value of a Local Variable can be unique to each different entity, this is useful if you have several entities of the same actor but you want their variable values to change independently.

The following program sets the Local Variable Damage to 0 when the entity is created. When the entity collides with another entity, its damage is increased by 1. Finally, the entity is destroyed when the Local Variable Damage is greater than 20. All other entities of the same actor are unaffected and won’t be destroyed.

If you don’t specify a condition for the If Statement, the code inside the If Body will be executed.

Else

In some situations, we want to take action when a condition is not met. For this, press the ELSE button.

Commands contained in the Else Body are executed when a condition is not met.

The program pictured above can be phrased in normal language:

If Player One’s Score is greater than Player Two’s Score, Player One wins!!! Otherwise, Player Two wins!!!

5.6 Control Flow

Topics:

• What is Control Flow?
• Making Decisions
• Iteration

5.6.1 What is Control Flow?

Within Actimator, commands are executed in sequence from top to bottom. There are two ways to specify the order in which commands should be executed, these are Control Flow Statements.

The two types of Control Flow are Conditionals(If Statements) and Iterations(For Loops).

5.6.2 Making Decisions

If

The If Statement checks a Condition. If the condition is met, it executes the commands located inside the If Body.

Else

In some situations, we want to take action when a condition is not met. For this, press the ELSE button.

Commands contained in the Else Body are executed when a condition is not met.

The program pictured above can be phrased in normal language:

If Player One’s Score is greater than Player Two’s Score, Player One wins!!! Otherwise, Player Two wins!!!

5.6.3 Iteration

You may need to execute commands repetitively. A For Loop steps in the amount given until a condition is met. Commands inside the loop are executed each step(iteration).

A loop can be broken up into these four parts:

Assignment

You assign an initial value to a variable at the start of the loop.
Condition

The loop will continue to step as long as its condition is met. For example, it will step until its variable reaches a certain value.

If the condition is not met, the loop will exit.

Loop Body

The commands in the Loop Body are executed each iteration.

Step

After the Loop Body, the variable will change its value and the loop will return to the condition.

The following program initializes the loop by setting the variable “x” to 0. As long as x is less than 5, the loop will create a Box Entity with a horizontal position of 32 * x. After creating the box, the loop completes its iteration by increasing x by 1. The loop then cycles back to the condition.

This program results in five boxes being created on the scene. Each one is created 32 pixels to right of the previous box.

Infinite Loops

You can make a loop that continues cycling the entire time the game is running. Loops can be demanding on your device’s processor, so be careful with loops whose condition will always be met.

5.7 Entity Management

Topics:

- Creating an Entity Programmatically
- Entity Parameters
- Getting This Entity’s Parameters
- Setting This Entity’s Parameters
- Storing an Entity in a Variable
- Getting an Entity’s Parameter using its Name
- Setting an Entity’s Parameter Using its Name
- Testing for a Specific Actor
- Destroying an Entity

5.7.1 Creating an Entity Programmatically

In its simplest form, the Create Entity Command to creates an entity at a specific position on the scene.

5.7.2 Entity Parameters

Entity Parameters have a value (number, string, boolean) that can be set or returned. Here’s a list of the different entity parameters:

- Name – Sets or returns the entity’s name
- Image – Sets or returns the entity’s image
- Top – Sets or returns the topmost point of the entity on the y axis
- Left – Sets or returns the leftmost point of the entity on the x axis
- Layer – Sets or returns the layer that contains the entity
- Image Width – Sets or returns the width of the entity’s image in pixels
- Image Height – Sets or returns the height of the entity’s image in pixels
- Image left offset – Sets or returns the entity’s image position relative to the entity’s actual center.
- Image top offset – Sets or returns the entity’s image position relative to the entity’s actual center.
- Rotation – Sets or returns the entity’s rotation
- Has Physics? – Sets or returns whether the entity has physics enabled
- Type – Sets the entity to a Static or Dynamic physics object
- Shape – Sets the shape of the entity to a Circle or Square for physics
- Collision Width Scale – Scales the entity’s collision area proportional to its image width
- Collision Height Scale – Scales the entity’s collision area proportional to its image height
- Friction – Sets the entity’s friction for physics
- Density – Sets the entity’s density for physics
- Restitution – Sets the amount that an entity bounces during a physics collision
- Fixed Rotation? – Sets or returns whether the entity’s rotation is constant
- Speed – Sets the speed of the entity
- Speed Direction – Sets the direction that the speed will be applied in
- Angular Speed – Applies torque to the entity

5.7.3 Getting This Entity’s Parameters

You can return the value of an entity parameter using the This Entity Command.

The returned value from the entity parameter can be
5.7.4 Setting This Entity’s Parameters

The values in entity parameters can be set or changed while the game is running. You use the same This Entity Command, you just set a value instead of returning it. Parameters can be set the same way as variables however, there are some restrictions.

- Name, Image, Type, and Shape can only be set to a string value
- Top, Left, Layer, Image Width, Image Height, Image Left Offset, Image Top Offset, Rotation, Collision Width Scale, Collision Height Scale, Friction, Density, Restitution, Speed, Speed Direction, and Angular Speed can only be set to a number
- Has Physics? and Fixed Rotation can only be set to boolean, 1(true) or 0(false)

Here are some ways you could set an entity parameter:

5.7.5 Storing an Entity in a Variable

The Create Entity Command returns a random unique Name(in the form of a String) of the entity it creates. This Name can be stored in a variable and later be used to change the created entity’s parameters.

5.7.6 Getting an Entity’s Parameter using its Name

Just like you can return the value in an entity parameter using the This Entity Command, you can return the value of any entity’s parameter using the Entity Property Command.

5.7.7 Setting an Entity’s Parameter Using its Name

With an entity’s Name you can set the value of it’s parameters. You can even change it’s Name.

5.7.8 Testing for a Specific Actor

You may want different things to happen when an entity collides with entities of different actors.

The program below is triggered at the start of every collision. The If Statement uses the Collided Actor Command to check if the entity is colliding with any entities of the Danger Actor.

5.7.9 Destroying an Entity

When an entity is no longer needed, it can be destroyed. You use the Destroy Entity Command and specify the entity that needs to be destroyed. You can use a variable, the This Entity Command, or type the name of the entity.

5.8 Asset Management

Topics:
- Changing the Image of an Entity
- Playing an Audio Asset
- Writing Text on the Scene

5.8.1 Changing the Image of an Entity

Actimator accepts .PNG, .BMP, .GIF, and .JPEG images. Image assets can either be used for Actors, or as backgrounds. It is best to use .png files for image assets.

The .PNG format:

PNG(Portable Network Graphics) is a widespread lossless image format that supports transparency. Thanks to compression, PNGs tend to have a very small file size.

Image assets are just for visuals, they can’t have a program or physics. You can drag an Image Asset directly
to the scene to be used as a background, by default, it will be placed on layer 1. While on the scene, we can control the size, position, layer, etc. of each image by clicking on it. The Image Asset Properties will appear in the Property Explorer.

Image boundaries are traced in purple with a purple dot on each side and corner. We can use these dots to resize each image.

An image can also be applied to an actor, all entities of that actor will have the same image. The entity parameters; Image, Image Width, Image Height, Image Left Offset, Image Top Offset, and Rotation control how an entity’s image looks.

Animation

Image assets can be used in sequence to create animations. There are many methods for creating animation. For this example, we’ll use the Tick Count.

This animation has 6 frames that we want to cycle through in the Before Render event. We have to make an If statement for each frame. We’ll check if Tick Count modulo 5 is equal to a number corresponding with each frame. For example, if Tick Count modulo 5 equals 0 then change image to frame 1.

5.8.3 Writing Text on the Scene

The Text Command displays text on the scene with the desired position, layer, font, etc.

- Text – The text you would like to display, this could be a String, Number, Mathematical Operation, Variable, or Parameter
- X – The X Position of the text
- Y – The Y Position of the text
- Layer – The layer for the text, -1 is the front-most layer
- Font – Pick a cool font
- Size – Size of the font
- Color – Color of the text
- Background – The background color of the text

The Text Command is also useful for debugging. The example pictured below displays the This Entity Speed parameter as text. You can compare parameter values with how entities behave, to improve the quality of your game.

5.9 Scene Management

Topics:
- Scene Properties
- Getting Scene Properties
- Setting Scene Properties
- Switching Between Scenes
5.9.1 **Scene Properties**

Scenes are divided into layers and contain the different elements that make up a game.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display</td>
<td></td>
</tr>
<tr>
<td>World’s Width</td>
<td>1280</td>
</tr>
<tr>
<td>World’s Height</td>
<td>720</td>
</tr>
<tr>
<td>Aspect Ratio</td>
<td>16:9</td>
</tr>
<tr>
<td>Scaling Method</td>
<td>No Scaling</td>
</tr>
<tr>
<td>Physics</td>
<td></td>
</tr>
<tr>
<td>Scale (pixels/meter)</td>
<td>30</td>
</tr>
<tr>
<td>Gravity - X</td>
<td>0</td>
</tr>
<tr>
<td>Gravity - Y</td>
<td>20</td>
</tr>
<tr>
<td>Allow Sleep</td>
<td>false</td>
</tr>
<tr>
<td>Collision Outlines</td>
<td>true</td>
</tr>
<tr>
<td>Layers</td>
<td></td>
</tr>
<tr>
<td>Layers</td>
<td>5</td>
</tr>
<tr>
<td>Default Entity Layer</td>
<td>2</td>
</tr>
</tbody>
</table>

Scenes, like entities, have their own properties:

- World’s Width – The width of the scene in pixels
- World’s Height – The height of the scene in pixels
- Aspect Ratio – The ratio of the scene’s size to the device’s screen
- Scaling Method – Either the scene has No Scaling or it’s set to Fit in Viewport
- Scale – The pixel-to-meter ratio of physics on the scene
- Gravity x & y – The amount of gravity (in the form of a vector) on the x and y axis for the scene
- Allow Sleep – Specifies whether or not physics objects can sleep
- Collision Outlines – Sets the physics’ bounding-box to be visible or hidden
- Layers – The maximum number of layers on the scene
- Default Entity Layer – The layer that entities are placed by default, this layer is set update automatically

5.9.2 **Getting Scene Properties**

You can return the value of a scene parameter using the Scene Property Command.

![Scene Property Command](image)

The returned value from the scene parameter can be stored in a variable, used in a mathematical operation, etc.

5.9.3 **Setting Scene Properties**

The values in scene parameters can be set or changed while the game is running. You use the same Scene Property Command, but you just set a value instead of returning it. Parameters can be set the same way as variables however, there are some restrictions.

- Id and Scaling Method can only be set to a string value
- Gravity x, Gravity y, Width, Height, Scale, Camera x and Camera y can only be set to a number
- Allow Sleep and Collision Outlines can only be set to boolean, 1(true) or 0(false)

Here’s a way you could set scene parameters:

![Setting Scene Parameters](image)

By setting the Camera X & Y parameters to the entity’s Top & Left, the entity will always remain in the viewport.

5.9.4 **Switching Between Scenes**

For games with more than one scene, you use the Switch Scene Command to move to any other scene.

![Switch Scene Command](image)

When the Switch Scene Command is executed, an instance of the other scene is created and the current scene is destroyed. Each time you move to a scene its entities execute their create events.
5.10 Physics

Topics:

- Physics Properties of an Entity
- Setting Physics Properties of an Entity
- Physics Commands
- Physics Properties of a Scene
- Setting Physics Properties of a Scene

5.10.1 Physics Properties of an Entity

You can set the physics properties of an actor so all of its entities will inherit them. You can also edit each individual entity’s physics properties using Entity Parameters.

- Has Physics? – Sets or returns whether the the entity has physics enabled
- Type – Sets the entity to a Static or Dynamic physics object
- Shape – Sets the shape of the entity to a Circle or Square for physics
- Collision Width Scale – Scales the entity’s collision area proportional to its image width
- Collision Height Scale – Scales the entity’s collision area proportional to its image height
- Friction – Sets the entity’s friction for physics
- Density – Sets the entity’s density for physics
- Restitution – Sets the amount that an entity bounces during a physics collision
- Fixed Rotation? – Sets or returns whether the the entity’s rotation is constant
- Speed – Sets the speed of the entity
- Speed Direction – Sets the direction that the speed will be applied in
- Angular Speed – Applies torque to the entity

Rotation:

Rotations start at 0°(pointing up) turning clockwise to 360°

Physics Units and Forces:

Actimator uses Box2D for physics. All physics units are metric and rely on the Scale Parameter(pixels per meter ratio).

- Gravity – Measured in meters per second per second
- Density – Measured in kilograms per meter squared
- Friction – The friction parameter is usually set between 0 and 1, but can be any non-negative value. A friction value of 0 turns off friction and a value of 1 makes the friction strong
- Restitution – Restitution is used to make objects bounce. The restitution value is usually set to be between 0 and 1. Consider dropping a ball on a table. A value of zero means the ball won’t bounce. This is called an inelastic collision. A value of one means the ball’s velocity will be exactly reflected. This is called a perfectly elastic collision.

This information is taken from the Box2D manual

5.10.2 Getting Physics Properties of an Entity

You can return the value of an entity’s physics property using the This Entity Command.

5.10.3 Setting Physics Properties of an Entity

You can set physics properties using the This Entity Command.
5.10.4 Physics Commands

**Impulse**

This command applies an impulse(hit) to the entity’s center of mass with the desired power and angle.

\[
\text{power} = 25 \\
\text{angle} = 45^\circ
\]

**Speed Direction**

This command applies a constant velocity to the entity’s center of mass with the desired power and angle.

\[
\text{speed} = 10 \\
\text{angle} = 270^\circ
\]

**Speed Vector**

This command applies a constant velocity to the entity’s center of mass with a power and angle that is determined by a vector.

\[
\text{speed} = 4.2 \text{m/s} \\
\text{angle} = 45^\circ \\
x = 3 \text{m} \\
y = -3 \text{m}
\]

5.10.5 Physics Properties of a Scene

These three Scene properties control physics:

**Scale**

This is the pixel-to-meter ratio of physics on the scene. By default this is 30, so an entity that’s 60 pixels wide is considered 2 meters wide for physics.

**Gravity x & y**

The amount of gravity (in the form of a vector) on the x and y axis for the scene. For example, a Gravity y of 5 would cause entities with physics to fall at a rate of 5 meters per second squared.

**Collision Outlines**

Sets the physics’ bounding-box to be visible or hidden. This is useful for debugging how physics entities collide with one another, but is not necessary.

5.10.6 Getting Physics Properties of a Scene

You can return the value of a scene’s physics property using the Scene Property Command.

In the example pictured below, we are returning the values of Gravity x and Gravity y to apply the magnitude of the vector to a variable.

5.10.7 Setting Physics Properties of a Scene

You can set the physics properties of the scene in the Property Explorer or you can set the value of a scene parameter while the game is running.

Here’s an example of setting the scene’s Gravity y value and turning off Collision Outlines from an entity’s program:

5.11 Leaderboard
6. Advanced Programming Topics

6.1 Display and Camera Management

Topics:
- Creating Large Scenes
- Cross-Display Settings
- Camera Management

6.1.1 Creating Large Scenes

6.1.2 Cross-Display Settings

6.1.3 Camera Management

6.2 Layers

Topics:
- What is a Layer
- Actimotor’s Default Layer Settings
- Updating Layers Programmatically
- Changing Layer Settings

6.2.1 What is a Layer

Layers allow you to lay images on the scene with depth. Layers with a higher number appear in front of others. When using layers, keep these concepts in mind:
- When we drag an image asset onto the scene it is automatically placed on layer 1
- The Default Entity layer is always updated
- The layer containing text is always updated
- Take care not to have more layers than is necessary
- Entities and images placed on non-existent layers will not be shown.
- Updating layers can be very demanding on our device, so only update when you need to
• -1 represents the front-most layer regardless of how many layers you have
• Entity physics are independent of layers. No matter what layer entities are on, they will collide with each other

6.2.2 Actimator’s Default Layer Settings

By default, Actimator has 5 layers (0-4). Layer 0 is the very back, Layer 4 is the front most layer unless we increase the number of layers. Layers remain transparent until something is drawn on them, these images are cleared each time a the layer is updated. On the layers that are automatically updated, the layer is cleared back to transparent 60 times per second. On manually updated layers, we can choose how often to update the layer.
6.2.3 Updating Layers Programmatically

A moving entity on the scene will leave a trail wherever it is drawn until the layer it’s contained in is updated.

When you need to update a layer, you can use the Update Layer command. You can type in the number of the layer, or you can even use the This Entity Layer Command.

If you wish to update all layers, use the Update All Layers Command. Updating layers takes up a lot of your device’s processing power, use this command with caution.

6.2.4 Changing Layer Settings

Your game may require more than the default number of layers. In the scene’s Property Explorer you can change the maximum number of layers. If you wish to have 4 layers for backgrounds(0-3), you can make layer 4 the default entity layer. If you then wanted 3 foreground layers(5-7), your total number of layers would be 8.
7. Publish Your Game

7.1 Publish Your Game with a Single Click

So you have made a game with Actimator. Congratulations! Now it’s time to share it with the world! In order to share your game, you have to Publish it. With Actimator, you can publish your game on multiple platforms including the Web, iPhone and iPad (iOS), and Android.

Your game will have a beautiful homepage including the game app, screenshots, story, tagline, etc. There is a Play button in the game’s homepage that allows page visitors to play the game. You can share the game’s homepage link with anyone and they can play your game and comment on it.

What is great about Actimator is that it automatically builds the Web, iOS, and Android apps of your game, with no extra effort on your side. Games published with Actimator are:

- Cross-Platform: Actimator supports the following platforms. For mobile platforms, Actimator generates the native app that can be uploaded and sold on their app store.
  - Web Publishing: Your game runs in the recent version of major Web browsers including Google Chrome, Mozilla FireFox, Apple Safari, and Microsoft Internet Explorer.
  - iOS Publishing: You will get the iOS app file of your game (the .ipa file). You can then publish your game on Apple’s App Store so that others can play your game on their iPhone and iPad.
  - Android Publishing: You will get the Android app file of your game (the .apk file). You can then publish your game on Google Play Store so that others can play your game on their Android devices.
- Cross-Device:
  - Web: You game runs on desktop, laptop, and mobile devices. However, for mobile devices, the native apps provide a better game play experience to players of your game.
  - iOS: The native app of your game can be uploaded to the Apple App Store and it runs on any iPhone, iPad, and iPad Mini, as long as your device is not too old.
  - Android: The native app of your game can be uploaded to the Google Play Store and it runs on most Android devices as long as your device is not too old.
- Cross-Display: No matter what the screen size of your device is, actimator scales the scenes up or down to give the same game play experience on any display size.

7.2 Quality Assurance

Before publishing your game, there are a few things that you can do to improve the final published game. These improvements lead to a better game that will increase the engagement of players and the final success of your game. In this page, we provide a set of guidelines to help you improve your game.
**7.2.1 Visual Design**

**Artwork Consistency**
- All artwork contained within in a game should adhere to the same styling. If you use a cartoon style for your game’s characters, then you should make the game’s world cartoon-like as well.
- Game characters and images should be sized in proportion to one another.
- Your game’s user interface should use the same font(s), color(s), style on all buttons and user interface.
- Use high resolution images (128x128 pixels is a good size for characters)

**Artwork Functionality**
- Colors should not be used in a way that makes it difficult for the player to distinguish what is going on.
- Game artwork should be easily visible on mobile devices (try not to use very small text)
- Game characters size and shape should match up nicely with their collision boundaries.

**Usage Rights**

Make sure you own all Intelectual Property (images, sounds, characters, etc.) before publishing your game.

**7.2.2 Game Mechanics Design**

**Controls**
- Make sure the controls for your game match nicely with target device (don’t use tiny buttons for mobile devices)
- Build controls that can be used by wide range of players.

**Gameplay**
- Make sure gameplay is easy to understand or provide players with a instructions.
- Your game shouldn’t be too difficult or too easy for your target audience.
- Your game should be free of errors.
- Most importantly, the game should be fun.

**Performance**
- Keep file sizes for assets to a minimum while maintaining quality. This will help speed up downloading the game to a device and shorten the load time for the game.
- Remove any unnecessary assets from you game.

**7.2.3 Debug Your Game**

Before you publish your game it should be free of bugs. Bugs (errors) are annoying to players and distract from the game.

**Play-test Your Game**

An easy way to catch bugs in your code, is to actually play your game. If there is a part of your code that you weren’t exactly sure how it would work, test everything related to that part. When play-testing, you have to try every possible thing that the player can do. If you encounter a bug, try to reproduce the conditions leading up to the bug and make changes to your code accordingly.

Even when your code runs exactly as intended, your game may still contain design flaws. An example would be the game is easy for you, but not for most players. Your game’s difficulty, controls, etc. should be designed to meet your target audience’s needs.
With play-testing, more testers are always better. Get as much feedback as you can from players of different ages, gender, etc.

**Fix bugs on step at a time**

When correcting your code, make a small change and test for the desired result. If you still have an error, change your code to how it was before and make a different change. You can keep repeating this process, making as few changes as possible so you don’t create another bug in the process.

### 7.3 Prepare Your Game Page

When you are ready to publish your game, set up your game page if you haven’t so already.

- Go to My Projects
- Click on the More drop-down menu and select Settings

![More dropdown menu with options: Publish, Settings, Duplicate, Delete]

1. Confirm the name of your game
2. Confirm the domain name of your game
3. Create a catchy tagline for your game
4. Write a story/description for your game
5. Upload a project icon and splash screen
6. Add an iOS App Link if your game is published on the App Store

Click on the screenshots tab:

Upload up to five screenshots.

### 7.4 Save a Version of Your Game

Before you can publish, you need to save a version of your game. When you save a version, the current state of your game (code, scenes, actors, assets) is saved separately from your project. A saved version of your game is unaffected by future changes made to your project. Once you have a version you can publish it as many times as you want to whichever platform you wish.

- You only need to save a version when you want to publish your game.
- The version number must be between 1.0 and 99.9 with exactly one decimal digit. Example: 1.0
- Start with the version 1.0 and increase the version number as you progress in the development.

Once you have a version of your game saved, you are ready to publish your game. From here, proceed to:

- Publish to Web
- Publish to iOS
- Publish to Android
7.5 Publish Your Game for the Web

Once you create a version of your game, you can publish your game for the Web. This will allow anyone to play your game on any device using a Web browser.

1. To publish your game to the Web, go to the Publish page:
   - If you are in the Lab, click on the Publish button, and then click on the Go to Publish Page link.
   - If you are in My-Projects page, find the game that you want to publish, click on the More button, and the click on the Publish button.
2. Once you are on the Publish page, click on the Web tab.
3. Read the instructions to publish your game.
4. Select the version of your game that you wish to publish from the drop down menu.
5. Press the Publish to Web button.

You have to see a message like “The new version was successfully published on the Web.” That’s All. Your game is published and You can share it on your social network to let others play your game.

- Your game page is located at: https://www.actimator.com/app/[your game’s domain]
- Your game can be played at: https://www.actimator.com/app/play/[your game’s domain]

7.6 Playtest on Smartphones and Tablets

You can play and test games made with Actimator on smartphones and tablets running iOS (iPhone/iPad) and Android. To do so, you will need to install the Actimator Player app from the App store on your device.

7.6.1 Playtest Your Game on Android

You can play and test games made with Actimator on Android devices. To do so, you will need to install the Actimator Player app from the Google Play Store.

Once you create a version of your game, to play it on an Android device:

1. go to the Publish page.
2. Go to Android tab.
3. Select the version of your game that you wish to publish from the drop down menu.
4. press the Publish to Android button.

Then go to your Android device:

1. Install Actimator Player from Google Play Store.
2. Open Actimator Player.
3. Enter the game domain.
4. Press Play.

The app loads your game and runs it as a native app on your device. A few things to notice:

- You can share your game domain with your friends and have them play and test your game, and give you feedback.
- Game Center features are not available on Actimator Player. They are only available on the published Android game.
- To load another game, you have to restart the player.

7.6.2 Playtest Your Game on iPhone/iPad (iOS)

You can play and test games made with Actimator on iOS devices. To do so, you will need to install the Actimator Player app from the Google Play Store.

Once you create a version of your game, to play it on an iOS device:

1. go to the Publish page.
2. Go to iOS tab.
3. Select the version of your game that you wish to publish from the drop down menu.
4. press the Publish to iOS button.

Then go to your iOS device:

1. Install Actimator Player from App Store.
2. Open Actimator Player.
3. Enter the game domain.
4. Press *Play*.

The app loads your game and runs it as a native app on your device. A few things to notice:

- You can share your game domain with your friends and have them play and test your game, and give you feedback.
- Game Center features are not available on Actimator Player. They are only available on the published iOS game.
- To load another game, you have to restart the player.
8. Leader Training

**Goal:** Equip you with skills to be able to lead a group of learners (your peers) through learning game design.

### 8.1 Peer-led Team Learning (PLTL) Orientation

Activator uses and supports the PLTL model to encourage teamwork and collaborative learning among peers.

#### 8.1.1 What is Peer Led Team Learning (PLTL)?

PLTL is an active, engaged learning experience for learners working collaboratively on challenging content while guided by a peer leader. It is learning through collaboration and participation.

![Image from PLTL International Society](image)

#### 8.1.2 How does it work?

Peer Leaders help guide learners using various learning strategies for problem solving, offering timely assistance when an individual or the group needs help and providing guidance as well as encouragement.
8.1.3 Roles

**Learner:** a student who embarks on a learning adventure in collaboration with peers

**Leader:** a peer who has successfully completed the course that they wish to lead, and will guide learners through the completion of active learning exercises in a peer-mentoring environment. Leaders are facilitators, not instructors, not answer-givers. They are coaches, facilitating with a “light hand” connecting learners to resources and other peers who can help.

8.1.4 Expectations

- PLTL is all about working together as a group.
- Peer leaders are trained to be facilitators, not lecturers or teaching assistants. They do not provide answers to learners; instead they guide them toward answers and set a tone for group discussion and learning.
- Learners do most of the explaining and reasoning with the rest of the group.
- Individual points of view and ideas are respected.
- Criticism is constructive.
- All members have an equal opportunity to participate.

8.1.5 What do leaders gain from PLTL?

- Increased confidence.
- Better understanding of the material and better performance in the material for themselves (Knowledge Building).
- Improved communication, critical thinking skills, and problem solving.
- Appreciation of value of collaborative learning and trust in the group problem-solving process.
- Community Service experience: (mentoring, helping fellow students)
- Educator experience
- Leadership Skill Building
These are skill sets frequently sought by employers.

8.1.6 Should you consider becoming a leader?

If you meet the following requirements and responsibilities:

• Understand and be fluent in content you will be leading.
• Be willing to learn and help others!
• Attend leader training *(already in progress!)*
• Participate in every session of next Cohort as a peer leader (6 sessions total; 2 times a week for 3 weeks)
• Participate in an interview after the sessions are completed

This is NOT for you if...

• You cannot commit to being at every session for entire upcoming Cohort.
• You do not care to be part of a group learning setting
• You do not want to prepare for the sessions.

8.2 21st Century Skills
The 21st century skills are a set of abilities that students need to develop in order to succeed in the information age. It includes **Learning and Innovation Skills** (Problem Solving, Critical Thinking, Creative Thinking, Collaboration, Communication), **Literacy Skills** (Information, Media & Technology Literacy, Digital Citizenship) and **Life and Career Skills** (Leadership, Flexibility, Initiative, Social Skills, Productivity).

In this session we will focus on a particular subset of these skills:

1. Digital Citizenship
2. Communication
3. Collaboration
4. Problem Solving
5. Leadership

**Resources**

P12’s Framework for 21st Century Learning

### 8.2.1 Digital Citizenship

Digital citizenship can be defined as the norms of appropriate, responsible behavior with regard to technology use. It is important to know how to be a good citizen in the digital realm. Provided below are some resources.

**Resources**

- Digital Citizenship Policy
- Digital Citizenship: Addressing Appropriate Technology Behavior
- Digital Citizenship: Using Technology appropriately
- Social Media in Education (Edutopia)

**Activities**

None
8.2.2 Communication

The ability to effectively communicate is probably one of the most important life skills. Communication is a two-way process. Simply put, effective communication is getting your message across to the receiver. It is the sender’s responsibility to make sure that the receiver gets the message and that the message received is understood.

It is not easy, and we all have had experiences where our communication failed or ran into a barrier. Barriers are behaviors or perceptions that prevent people from communicating effectively, like negative attitudes or lack of motivation to communicate. Barriers keep us from understanding other’s ideas and thoughts and can appear at any point.

In order to be an effective communicator, you must monitor the effectiveness of your communication at all times and adjust your strategies to increase understanding, overcome barriers, and respond in an appropriate way.

Watch Me!

Tools and strategies
• 5W (Video)
  • Reflection - key ingredient to move knowledge from short-term to long-term memory
  • Focused, active listening - concentrating on what is being said
• Clarifying - checking for understanding
• Validating emotions recognizing and acknowledging others feelings
• Powerful Questions higher order thinking
• Breaking Resistance / Barriers: fear, assumptions, attitudes, limiting beliefs, resources and time, self-sabotage, lack of motivation

Activities

• Discussion - Suggested questions:
  – Why do you think it’s important to know how to communicate effectively?
  – What does active listening do for you? What does it require?
  – What are characteristics of effective communicators? How about ineffective communicators?

8.2.3 Collaboration

Collaboration is the process of two or more people or organizations working together to realize shared goals. Collaboration is very similar to cooperation. Most collaboration requires some form of leadership. As teams engage in collaborative problem solving, collaboration requires communication and encourages reflection on behavior.

Collaborative learning is based on the model that knowledge create is social - that is, knowledge can be created within a
Collaborative learning is manifested when groups of students work together to search for understanding, meaning, or solutions or to create an artifact or product of their learning. For example, in our Actimator sessions, learners and leaders work together to learn game design (and computer science concepts) and create a game. Unlike individual learning, people engaged in collaborative learning capitalize on one another’s resources and skills (asking one another for information, evaluating one another’s ideas, monitoring one another’s work, etc.).

Watch Me!

Watch the video for this step: https://player.vimeo.com/video/161115298

Tools & Strategies

- Round robin, popcorn discussions
- Think-pair-share (video): Partners/buddies (two learners alternating)
- PALS (peer assisted learning strategies): partnering strong with less strong learners
- Think aloud (video): modeling strategy where instructor/leader makes thought process explicit
- Cognitive scaffolds - discussions, questioning, brainstorming to build learning
- Collaborative Learning resources - learning is profoundly social

Activities

- Discussion questions:
  - Ultimately, we are all responsible for our own learning. Then why is collaborative learning important?
  - Any thoughts about the importance of collaboration in general?

8.2.4 Problem Solving

Problem Solving involves finding solutions to problems. In the context of game design in a group setting, that can mean anything from resolving technical problems (debugging) to resolving conflict between team members.

Tools & Strategies

- Critical thinking is exercising or involving careful judgment or evaluation, e.g., judging the possibility of an idea or product. Critical thinking is a very important skill to develop as. The ability to think critically is a valuable tool.
- Creative thinking is having unusual ideas and innovative thoughts, able to put things together in new and imaginative ways. It is like thinking outside of the box.
- Finding / Getting Help: It is important to keep in mind that all learners have trouble understanding things sometimes. If this happens to you, make sure you get the help you need right away!
  - If you are having trouble understanding – reread slowly, organize, review, and search for information.
  - If that doesn’t work, ask someone (teacher, leader, peer/classmate). Working with someone else will help you achieve more than you would be able to alone.
Knowing when and how to ask questions might be the most important thing you’ll ever learn. Not only will it help you learn, asking questions will help you accomplish your goals and feel good about yourself because you will gain insight, understanding, and skills that you can use in all areas of your life. When seeking for help, ask questions that will give you hints to get “unstuck” when you are stuck. Asking questions that start with “why?” or “how?” are good ways to do this. Remember the 5W1H strategy!

Activities

- Trouble-shooting activities in Actimator: For each challenge, run the game to find what the problem is and a solution to fix it. **Important:** Make sure you are logged in before clicking on the challenge links below.
  - Challenge 1 Hint: The asteroid is falling down.
  - Challenge 2
  - Challenge 3 Hint: The bird is not jumping when I click on the scene.
  - Challenge 4
  - Challenge 5
  - Challenge 6
  - Challenge 7

8.2.5 Leadership

Envision, Enable, Empower, Energize!

- Setting direction, vision, goals
- Using management skills to guide people to the right destination, in a smooth and efficient way
- Motivating and inspiring people to engage,
- Creating positive, open and collaborative environment to keep teams working effectively together
- Coaching and guiding teams, while instilling a sense of autonomy within their teams
- Dealing with wide variety of diverse audiences effectively, while making everyone feel important: utilizing each person’s unique skills and maximizing potential
- Embracing change
- Communicating clearly, concisely, and frequently – and engaging in two-way communication

Tools & Strategies

- **Tools** - understand **how, when, and why** to deploy tools at different parts of the process.
- **Organization** - create structure from chaos by using tools such as plans, schedules and checklists
- **Time Management** - prioritize, respect other people’s (teammates’) time, facilitate efficient discussions, change
- **Communication** is key! - already covered
- **Problem-Solving** abilities - already covered
- **Assessing Emotions**: a good leader reads the situation and the people's state of mind and reacts accordingly.
Activities

• Assessing and Reacting to Situations
• Discussion Questions:
  – What are the characteristics of good leader? (popcorn)
    * Effective leaders tend to be both likeable and assertive - confident, self-assured, but also humble
  – What are different forms of leadership? How does peer leadership differ?

Resources

Leadership guide

8.3 Learning Methodologies Resources

• Differentiation, different learning styles - may need to change method to meet learners needs
  – visual, auditory, kinesthetic (but be careful not to pigeonhole people into a single category)
  – use different models: learners are not expected to be clones of instructor’s/leader’s particular way of thinking
• Constructivism - knowledge is constructed through learners
• Inquiry-based learning - powerful questions
• Problem-based learning - challenging problems - collaboratively work toward resolution
• Knowledge Structures
  – Bloom’s taxonomy

8.4 Concluding Activities

• Breakout sessions
  – Share one thing you learned. Respond with a word (popcorn)
  – What are some concerns about being a leader? (round robin or think-pair-share)
    * What if I don’t know something?
    * What if nobody says anything?
  – Come up with some other concerns & solutions
• Exit Ticket
Index

A
actor ............................................. 17
asset ............................................. 15

E
Entities ........................................... 19

P
physics ............................................. 20
program ........................................... 18

S
scene ................................................. 19