GDevelop

Moha Jain

June 4, 2019

Abstract

GDevelop currently is an open-source game development software, allowing to create HTML5 and native games without any knowledge in a specific programming language. I plan to do the following changes.

- Write a new tutorial explaining step by step how to create a game.
- Improve the wiki pages.
- Improve extension developer guide.
- Documenting for publishing GDevelop on various platforms.

Contents

1 About Me

The Student

Name Moha Jain
e-mail mohajain99@gmail.com
Github mohajain

The Institute

University Indian Institute of Technology(BHU), Varanasi
Major Industrial Chemistry
Year Sophomore

1.1 Programming and Writing Experience

I have been programming for last two years in C++. Apart from C++, I have experience in HTML5, CSS3, Bootstrap4 (*a website built by me as a freelancer*)
Django, Python (*My Blog*)
React, JavaScript (*Tic Tac Toe game*).
Android, Java (*LoginApp*), (*DiceRollerApp*), (*CalculatorApp*).
For version control, I have been using git. I work on *macOS Mojave* Vesion 10.14.2 with *Sublime Text* as my primary editor.
I have done a course on technical writing offered by *Moscow Institute of Physics and Technology.* I have documented my *Exploratory Project* which took 6 months to complete. It involved various softwares like MAPS(*Materials and Process Stimulation*) by Scienomics, MCCS TOWHEE(*Monte Carlo for Chemical Stimulation*).

1.2 My Motivation

Building games on GDevelop is a favourite feature to me. I frequently use GDevelop to build innovative and challenging games which enhance my creativity and help me relax after a tiring day. I have always been fascinated about the technologies and algorithms used to build GDevelop. We can make different objects with the help of GDevelop Piskel Editor and can use them as background, and give it a behaviour like platform, platform character, top-down movement, draggable object etc.
Documenting it properly is a way to explore GDevelop even more.

1.3 Contributions

I started using GDevelop in April and I found many bugs in the README and *wiki* pages.

1.3.1 Merged Pull Request

1085 - Fixed typo in README

1.3.2 Issues/Bugs Caught

Wiki Page GDevelop:

First Paragraph: *How to make a platform game.*

This tutorial will help you to begin to use GDevelop: You will create a very simple platformer game where the player can jump on platforms and collect coins. Note that you can read the Getting Started page to get an overview of the software: it explains the main concepts and describes the interface of GDevelop.

Readability may be enhanced by removing the highlighted words.

Corrected First Paragraph

This tutorial will help you use GDevelop: You will create a simple platformer game where the player can jump on platforms and collect coins. Note you can read the Getting Started page to get an overview of the software: it explains the main concepts and describes the interface of GDevelop.

Second Paragraph: *Download GDevelop.*

If you do not have GDevelop 5, download it from the official website. It's available for Windows, macOS and Linux. Always download GDevelop from this page to be sure to have the latest version. Install (or extract) GDevelop and launch it.

Add a comma: ,and

Corrected Second Paragraph

If you do not have GDevelop 5, download it from the official website. It's available for Windows, macOS, and Linux. Always download GDevelop from this page to be sure to have the latest version. Install (or extract) GDevelop and launch it.

Fourth Paragraph: *Add a scene.*

Scenes are parts of your game. Typically, each different screen of your game is a scene: the main menu, the pause menu and the levels are scenes. Scenes contain objects, that are displayed on the screen. These objects can be put on the scene to create the levels (these objects are called "instances"). Scenes also contain the events that are run to animate the scene. In the project manager that is opened on the left, click on the + button under "Scenes": A new scene, called "New scene", is added to the list. Click on it to open it.

Passive verbs make writing less direct. We should try to use active verbs instead.
possible unnecessary comma.

Corrected Fourth Paragraph

Scenes are parts of your game. Typically, each different screen of your game is a scene: the main menu, the pause menu and the levels are scenes. Scenes contain objects that are displayed on the screen. We can put these objects on the scene to create the levels (we call these objects "instances"). Scenes also contain the events that are run to animate the scene. In the project manager that is opened on the left, click on the + button under "Scenes": A new scene, called "New scene" is added to the list. Click to open it.

Fifth Paragraph: *Create the player object.*

The player will be controlling a character that can jump and move on platforms. We're going to create this object. On the right, you can see a panel called "Objects". Click on the + button to add an object. A window will pop up displaying different types of objects you can choose from. Different kinds of objects provide specific features. For most of our objects, we will use the Sprite objects, which are animated objects that can be used for a lot of elements (player, platforms, enemies, items...). Click on Sprite in the list.

The new object is added to the scene and the object editor is opened: The object is empty for now. Sprite objects are composed of animations, and each animation can contain one or more images. Let's add an animation! Click on the + button. The animation is added but still empty with no images. To add an image, click on the + in the blank thumbnail.

Passive verbs make writing less direct. We should try use active verbs instead.
Readability may be enhanced.

Corrected Fifth Paragraph

The player will control a character that can jump and move on platforms. We will create this object. On the right, you can see a panel called "Objects." Click on the + button to add an object. A window will pop up displaying different objects you can choose from. Different objects provide specific features. For most of our objects, we will use the Sprite objects, which are animated objects that can be used for a lot of elements (player, platforms, enemies, items...). Click on sprite in the list. We add the new object to the scene and open the object editor. The object is empty for now. Sprite objects are composed of animations, and each animation can contain one or more images. Let's add an animation! Click on the + button. The animation is added but still empty with no images. To add an image, click on the + in the blank thumbnail.

Sixth Paragraph: *Put the object on the scene*

To display the player in our level, we place it in the scene. Each object we add to the scene is called an "instance" of the object. Our player instance will be unique as we are going to have only one in our level, but other objects (like platforms, coins, enemies...) can have multiple instances living on the scene. To add the object to the scene, simply click-and-drag the object from the list to the scene.

Readability may be enhanced.

Corrected Sixth Paragraph

To display the player at our level, we place it in the scene. Each object we add to the scene is called an "instance" of the object. Our player instance will be unique as we will have only one in our

level, but other objects (like platforms, coins, enemies...) can have multiple instances living on the scene. To add the object to the scene, click-and-drag the object from the list to the scene.

Eighth Paragraph: *Add platforms*

The level is still empty for now without any platforms. Let's add some. Create a new object by clicking on the + button in the Objects list. This time, choose Tiled Sprite when asked for the type of object: The object editor is opened. It's a bit different compared to the editor of the Sprite object, as Tiled Sprite objects are not animated. Click on Select an image and then on Choose a new image. This time, choose the image called grassHalfMid. Close the editor by clicking on the "Apply" button in the bottom right corner. Click on the object in the list to select it, and click on the scene to place it in the level: The object is added to our scene but it is very tiny. Click on it and use the handles to resize so that it looks like a platform: Also, don't forget to rename the object to be called "GrassPlatform" instead of the default name.

We might need a comma before **'but'** conjunction if it joins two independent clauses.
Passive verbs make writing less direct. We should try use active verbs instead.
Readability may be enhanced.

Corrected Eighth Paragraph

The level is still empty for now with no platforms. Let's add some. Create a new object by clicking on the + button in the Objects list. This time, choose Tiled Sprite when asked for the type of object: We open the object editor. It's different compared to the editor of the Sprite object, as Tiled Sprite objects are not animated. Click on Select an image and then "on Choose" a new image. This time, choose the image called grassHalfMid. Close the editor by clicking on the "Apply" button in the bottom right corner. Click on the object in the list to select it and click on the scene to place it at the level: We add the object to our scene, but it is very tiny. Click on it and use the handles to resize so that it looks like a platform: Also, don't forget to rename the object to be called "GrassPlatform" instead of the default name.

Tenth Paragraph: *The Player*

The movement of the player could be created using the events of GDevelop, but it would be a quite long and difficult task to recreate a good platformer engine. Luckily, GDevelop already comes with a complete platformer engine built-in that we can use. For this, we need to use the "platformer behaviors", which allow us to access and use the built-in platformer engine. Open the "Player" object editor: Switch to the Behaviors tab and click on the + button to add a behavior to the object. Choose Platformer character in the list of available behaviors. You can then see the behavior in the editor, with a lot of parameters that you can tweak: You can change some properties like the jump speed. Set it for example to 800 so that the jump is more powerful.

Passive verbs make writing less direct. We should try use active verbs instead.
Readability may be enhanced.

Corrected Tenth Paragraph

We can create the movement of the player using the events of GDevelop, but it would be a long and difficult task to recreate a good platformer engine. Luckily, GDevelop already comes with a complete platformer engine built-in that we can use. For this, we need to use the "platformer behaviors," which allow us to access and use the built-in platformer engine. Open the "Player" object editor: Switch to the Behaviors tab and click on the + button to add a behavior to the object. Choose platformer character in the list of behaviors. You can then see the behavior in the editor, with a lot of parameters you can tweak: You can change some properties like the jump speed. Set it for example to 800 so that the jump is more powerful.

Eleventh Paragraph: *Platforms*

We also need to set objects to be platforms, so the player can walk and jump on them. Otherwise, the player will fall through the object as we didn't tell to our character what to do when they land on it. By setting an object to be a platform, the platformer character will collide with it and be able to walk on it. To set the GrassPlatform object to be a platform, open the GrassPlatform object editor. Switch to the Behaviors tab, then add a behavior: this time, choose the Platform behavior. There are also a few options that you can change. For example, the option "Can grab platform ledges" is there to allow the platformer character to grab the edge of the platform. If we enable this option right now, the result will look strange though, as we don't have an animation to be displayed when the character grabs the edge. Let's keep this unchecked.

Passive verbs make writing less direct. We should try use active verbs instead.
Readability may be enhanced.

Corrected Eleventh Paragraph

We also need to set objects to be platforms, so the player can walk and jump on them. Otherwise, the player will fall through the object as we didn't tell our character what to do when they land on it. By setting an object to be a platform, the platformer character will collide with it and be able to walk on it. To set the GrassPlatform object to be a platform, open the GrassPlatform object editor. Switch to the Behaviors tab, then add a behavior: this time, choose the Platform behavior. There are also a few options that you can change. For example, the option "Can grab platform ledges" is there to allow the platformer character to grab the edge of the platform. If we enable this option, the result will look strange though, as we don't have an animation to display when the character grabs the edge. Let's keep this unchecked.

2 The Project

In this section, I will explain my project details and its vision. I expect this structure to be considerably enhanced under the guidance of my mentor.

2.1 Brief Overview

GDevelop is one of the few game development software, which allows anyone to create games. One of the finest features of GDevelop is that it is easy to use. But when it comes to its documentation, it has some major issues which are as follows:

- There are people who want to know the proper functioning of GDevelop, like what's going on behind the scenes, how it is working, etc. GDevelop provides no such information.
- It does not have a proper error-free wiki page, which is up to the standards of GDevelop.
- There should be some examples of a fully built game, through which we can show every function of GDevelop properly.

My main concerns for this project are:

- Write a new step-by-step tutorial explaining how to create a game.
- Improve the existing documentation by checking that the documentation covers every object, event, behavior, and by fixing the bugs and typos.

- Improve the documentation on GitHub to explain how to create a new extension for the software.
- more step-by-step tutorials and detailed explanations about marketplaces.

2.2 The Plan

As work done by the developers results that now GDevelop has a proper documentation for the beginners, I want to continue their work and want to complete the following:

- A brand new tutorial, without any bugs or typos.
- Improved wiki pages.
- Improved extension developer guide.
- Documentation and tutorial for publishing games on various platforms.

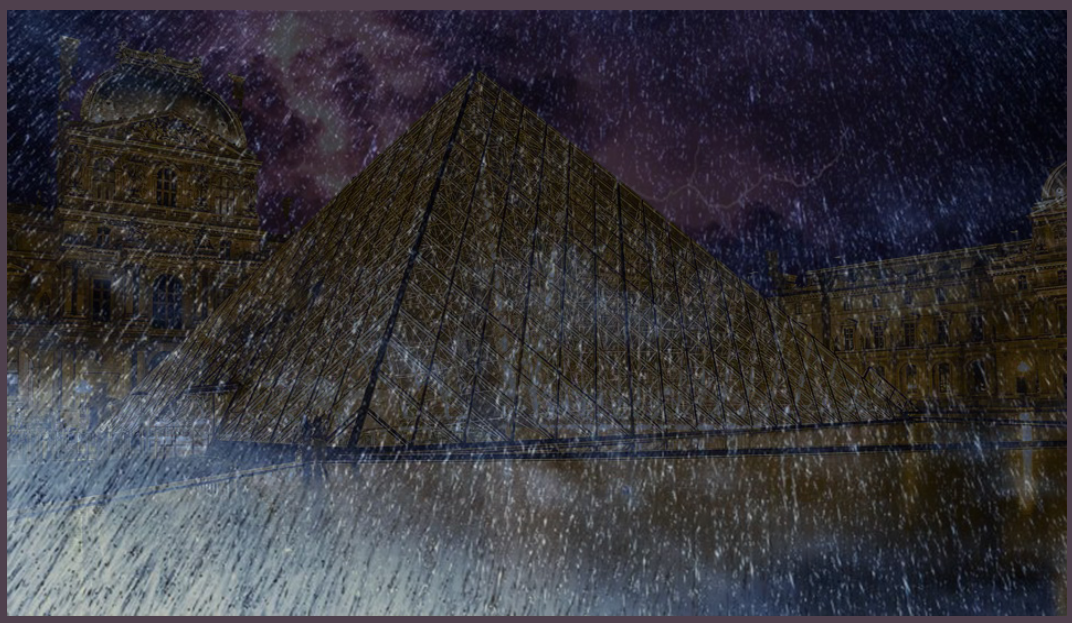
I have written a plot for a sample game and I have briefly given the details of the tech. which will be used to develop that game. I will discuss this plot with my mentors and I will make changes, if required, to make it better.

Santino D'Antonio

Written by - Moha Jain

June 2019

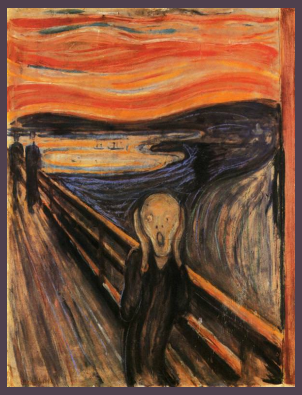
The Plot



Suddenly it started to pour. Within moments, the entire place came into the grasp of darkness and the black clouds began to roar. The lightning passed through the glass pane of Louvre Museum, *Rue de Rivoli*, 75001 Paris, France, which hosted the most highly anticipated exhibition of the world. It showcased 6 most expensive paintings ever made.

Santino D'Antonio, the most famous thief of the world, was setting his eyes on the paintings and it could be the biggest robbery ever committed.
He was preparing for this day for almost 3 months. He planned everything perfectly. He stole all the blueprints of the museum, and he exactly knew each and everything about the paintings. There were 6 paintings in the museum:

Room 1
The Scream - *Edvard Munch*



Room 2
The Persistence of Memory - *Salvador Dali*

Room 3

Guernica - *Pablo Picasso*

Room 4

The Last Supper - *Leonardo Da Vinci*

Room 5

Salvator Mundi - *Leonardo Da Vicnci*

Room 6

Mona Lisa - *Leonardo Da Vinci*

The security systems were flawless. If someone tried to enter the room at night and cross the hidden lasers, then high decibel alarms would ensure that every armed guard comes in action to catch the thief. And even if someone manages to escape them, and reach the painting, they will not be able to take it, as every painting is 13 feet high from the ground, with a deep cavity between the wall and ground. Moreover, there were green lasers too, which had a voltage drop of 10,0000V across them. Santino had to save himself from them.

As Santino D'Antinio was one of the most dangerous thieves out there, he planned everything beforehand. One of his friends was guarding the Louvre Museum that night. If he helped Santino, then in return, Santino would give him his share. He helped Santino by hiding guns in every room of the museum, so that Santino would kill the guards. He also kept a short-range jet pack in each room, so that Santino could take the painting off the wall. Each gun has one

bullet in it; that's why Santino cannot afford to waste any bullet. The jet pack must also be used wisely, because it has limited fuel.
Santino would be successful if he can steal all six paintings.
There will be a hidden level in this game, in which Santino has to leave the museum and reach his secret residence.

Tech and Tools Required

We will use **Objects** to create Santino, floor, gun, jetpack, and paintings. We will use the piskel editor to create the animations. If there are some complicated animations like a gun and a jet pack, then we can add them from our computer.
We will add empty **Events** and write conditions and the actions that will take place. For example, if we click the space bar, then Santino should jump. If we click shift, Santino must fire etc.
We will then add **Behavior** to our objects.For example, if we apply platformer character behavior to Santino, we'll see that it falls. Similarly, if we apply platform behavior to the floor, then Santino would fall on the floor, move on the platform, etc.

3 Timeline

3.1 Community Bonding Period

- Discuss the project with my mentor in further detail.
- Get more acquainted with GDevelop codebase and its features.
- I will have meetings with the mentors and we will come up with more ideas.

3.2 Documenting Period

- **Week 1,2,3 (September 2 - September 23)**

Write a brand new tutorial explaining how to create a game.

Make the existing tutorials error-free.

Catching up the pace.

- **Week 4,5,6 (September 24 - October 15)**

Improve the existing documentation by checking that every object, event, behavior is covered by the documentation and improve each page with new examples, links and explanations.

Make the existing pages error-free.

- **Week 7,8,9 (October 16 - November 6)**

Main focus on the extensions. Extensions are an important part of GDevelop. Improve the documentation on Github to explain how to create a new extension for the software.

Catch up any previous work left.

Fix bugs and make the existing *wiki* page for extension error-free.

- **Week 10,11,12 (November 7 - November 28)**

Create more documentation about publishing games on marketplaces, gaming websites and app stores.

Fix bugs and make the existing *wiki* page error-free.

Implementing new ideas which may have occurred recently, after discussing with mentors.

Finalizing the project report after discussing with my mentors.

Wrap up.

3.3 Post GSoD

If there is some debugging or any other thing left to do, I will try to complete it post GSoD and I will continue to give my contributions to GDevelop and help the new contributors. GDevelop provides me with a good platform providing intellectually challenging work in field of computers, for enriching my knowledge and skills.

References

GDevelop/4ian