Automotive Level Editor Docs

Inhoud

Scenes

Menu Scene

Description

A canvas with clickable text areas, on top of a background. This is the first scene you encounter. This scene sends you to the other scenes.

Runtime

Figure 1: Menu Scene in runtime, the button with "Spelen" goes to the Play Scene and the button with "Level Editor" goes to the Level Editor Scene.

Scene Construction

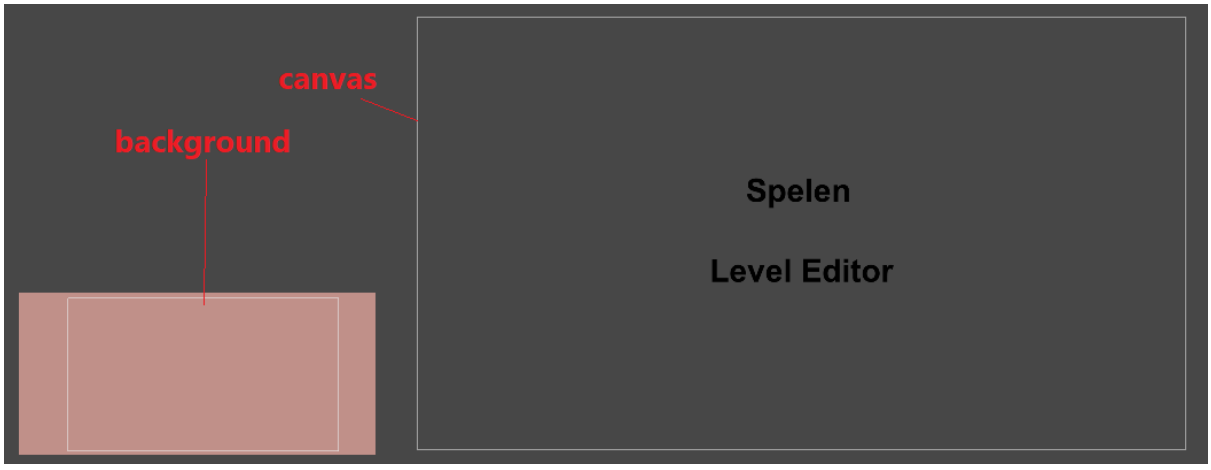


Figure 2: Menu Scene in the Unity Editor. The canvas is displayed on top of the background.

Play Scene

Description

The scene wherein you can play levels and **Car-Tasks** (minigames). Currently you can only load in levels you created in the same session inside the **Level-Editor Scene**.

***Car-Task**: A small minigame wherein you do a specific automotive task.*

***Level-Editor Scene**: The scene wherein you can create a level.*

Runtime

Figure 3: Play Scene in runtime

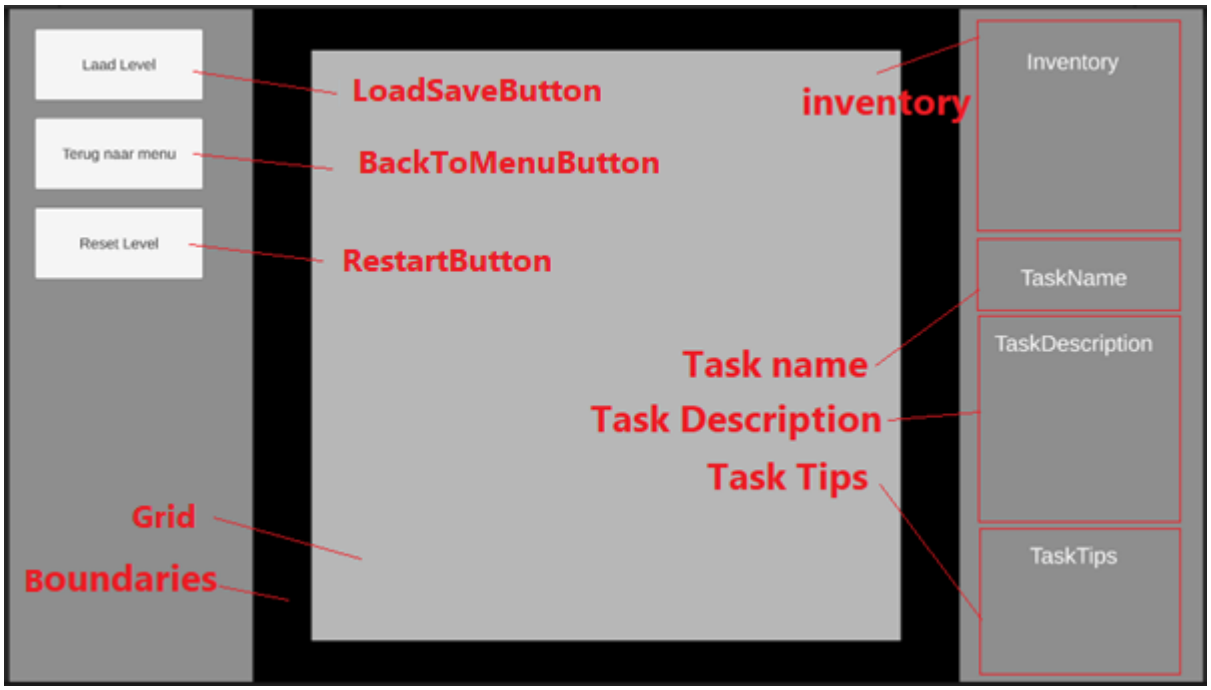


Figure 4: Play Scene in runtime with terminology

LoadSaveButton: loads in the saved level. That means all GameObjects and all assigned tasks.

BackToMenuButton: Sends you back to the Menu Scene.

ResetButton: Reloads the scene.

Grid: Doesn't add anything to the scene.

Boundaries: The Boundaries of the level. The player can't pass this GameObject.

Inventory: Here it will display all the items the player has picked up.

TaskName: Displays Current CarTask name.

TaskDescription: Displays Current CarTask description.

TaskTips: Displays Current CarTask tips. (currently not used)

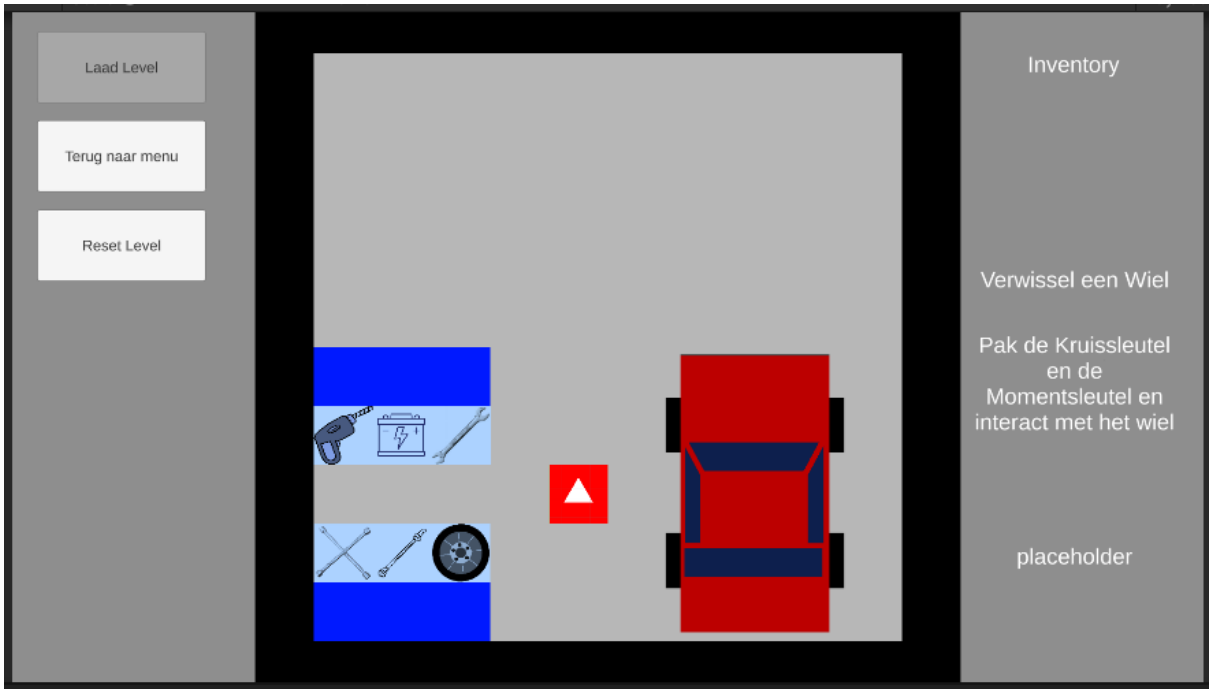


Figure 5: Play Scene in runtime after loading the level that has been created in the Level-Editor Scene

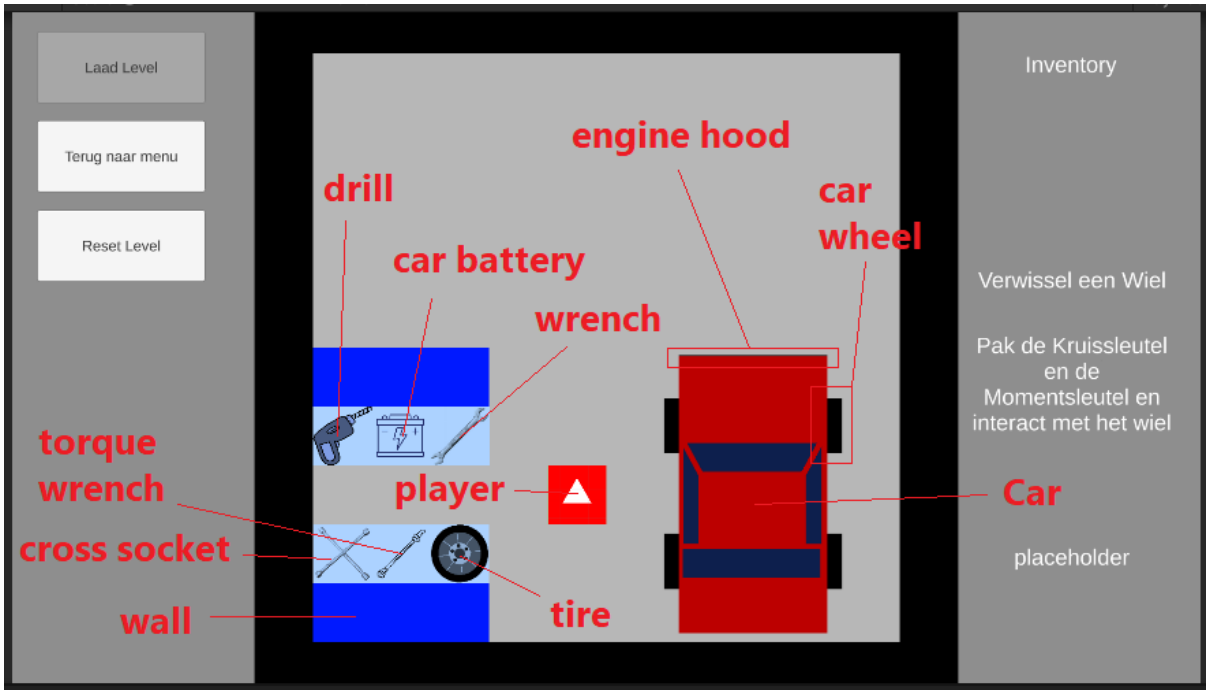


Figure 6: Play Scene in runtime after loading the level that has been created in the Level-Editor Scene with terminology. More of these items in the section Scripts > Level Editor > Item.

Scene Construction

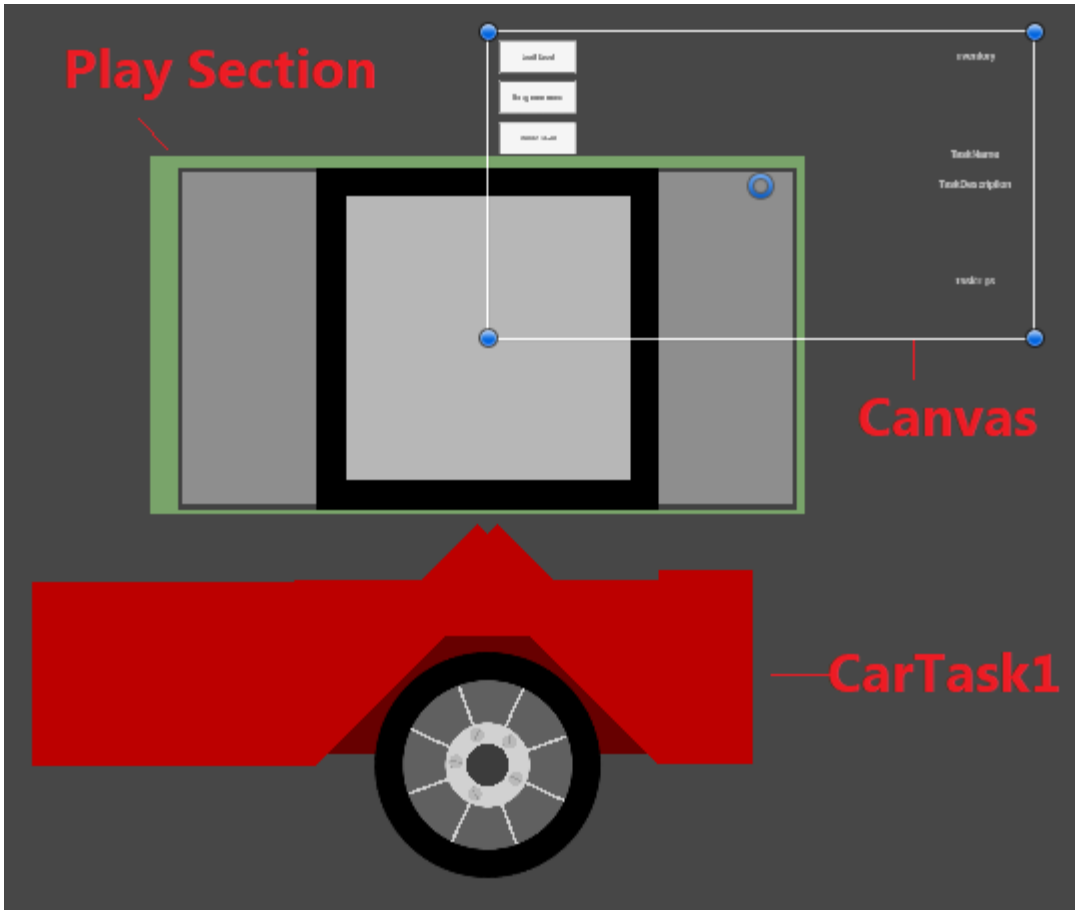


Figure 7: Play Scene in the Unity Editor

CarTask1: The first CarTask. Inside this task you remove a wheel and attach a new one to the “car”.

Level-Editor Scene

Description

Inside this scene you can create a level. This level can later be loaded inside the **Play Scene**.

***Play Scene**: The scene wherein you can play levels*

Runtime

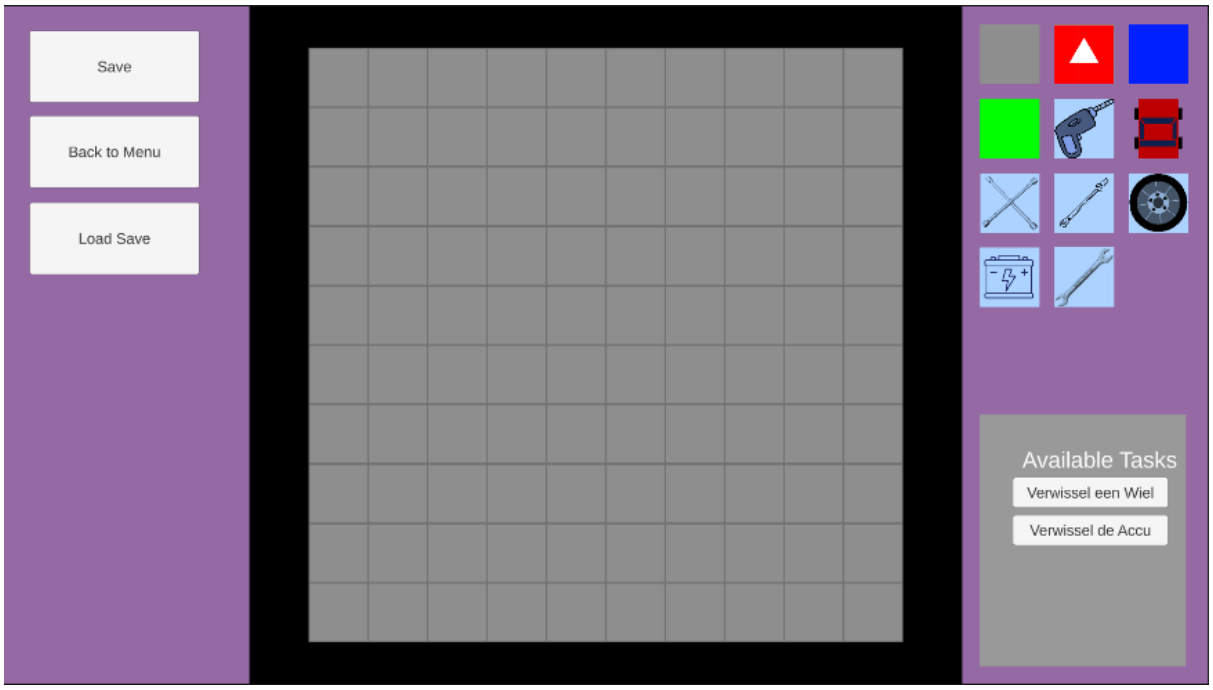


Figure 8: Level-Editor in runtime

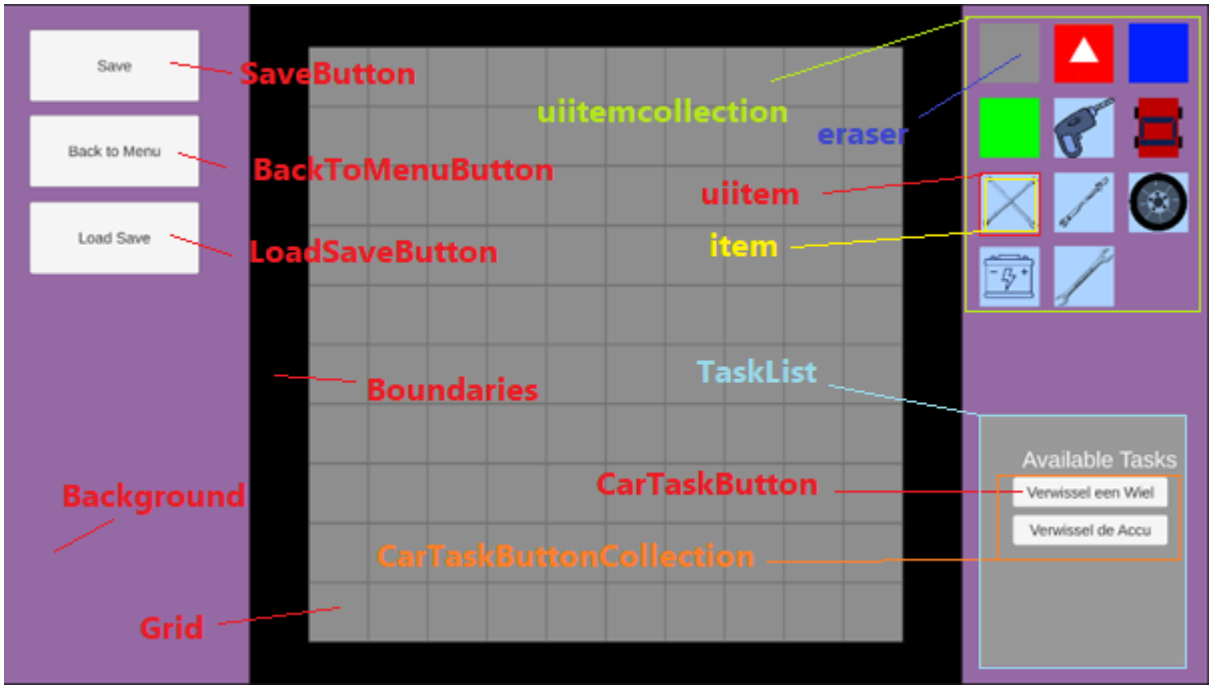


Figure 9: Level-Editor in runtime with Terminology

SaveButton: This button does nothing, everything is being saved automatically to the TextFiles Grid.txt and CarTask.txt.

BackToMenuButton: Brings you back to the menu.

LoadSaveButton: Loads all GameObjects and CarTasks from the TextFiles Grid.txt and CarTask.txt respectively.

Boundaries: Doesn't serve a use in the Editor besides making it look like the play section.

Background: Just a coloured rectangle

Grid: A grey rectangle with quite some logic in it. Find more about Grid in the section Scripts>Level Editor>Grid

UIItemCollection: Collection with all UIItems. The UIItems are hardcoded in the scene but this can all be done automatically.

Eraser: When clicked on the eraser you can click on an item on the grid to remove it. Eraser is the only item inside the UIItemCollection which is not inside a UIItem.

UIItem: An UI clickable box inside UIItemCollection which carries an prefab and an item. When Clicked on this UIItem, the prefab inside will be your selected prefab. If you click on the grid now, it will instantiate that prefab

Item: No use in the Level Editor.

CarTaskButton: You can assign a CarTask to your level by clicking on this button. When clicked it becomes green. When clicked again, it will be removed from the level and the button colour will turn back to white.

CarTaskButtonCollection: Collection of all the CarTaskButtons. This is all prepared in runtime, and scalable unlike the UIItemCollection.

Scene Construction

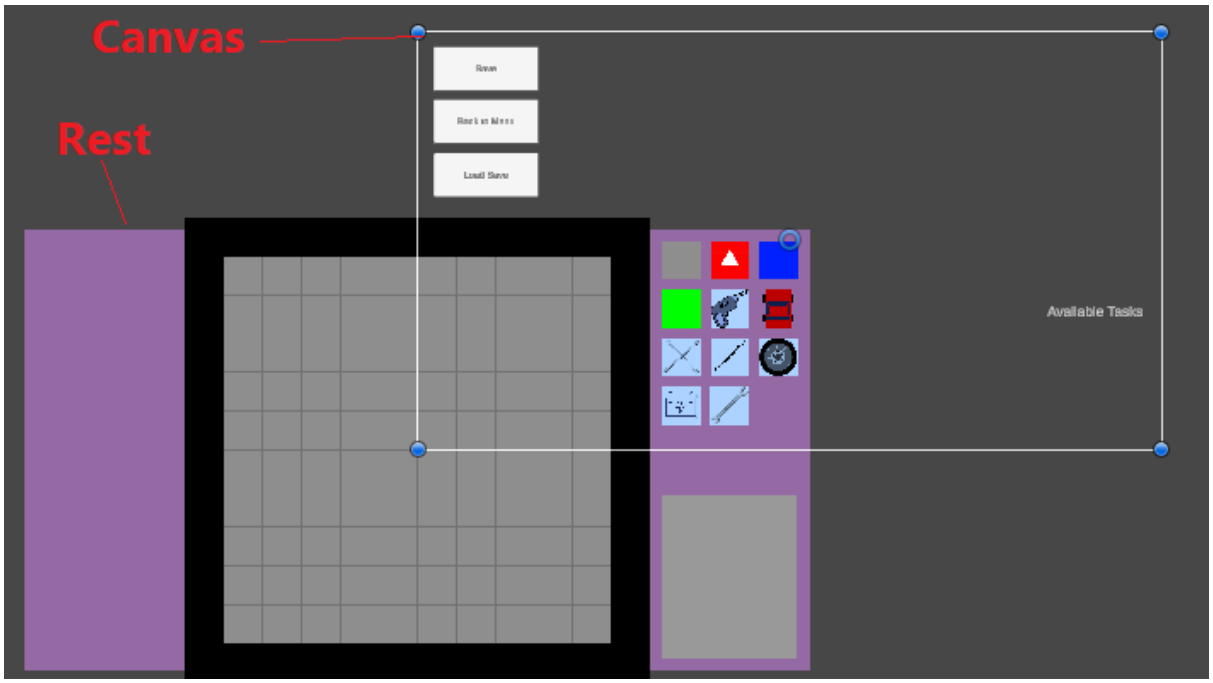


Figure 10: Level Editor inside the Unity Editor

Scripts

The Scripts have been sorted in 4 different folders, and there are also some scripts in the root. Let's go over the scripts

Root

Data

In this class you can find everything that has to do with writing data, formatting data and reading data. This class contains two other classes. `GridData` and `CarTaskData`

GridData

GridData is filled with static functions that write, read and format data. All these functions are static so that you can access them without having to make an instance. This class saves everything to the TextFile "Grid.txt". This class can be accessed from everywhere like the following: `Data.GridData`

CarTaskData

CarTaskData is filled with static functions that write, read and format data. All these functions are static so that you can access them without having to make an instance. This class saves everything to the TextFile "CarTasks". This class can be accessed from everywhere like the following: `Data.CarTaskData`

GameModeManager

This class is used to change the current gamemode. For example when you go from the Menu to the Editor, the following code runs: `GameModeManager.SetGamemode(Gamemodes.Level_Editor)`. This class can be accessed from everywhere and the function and property both are static so that you don't need to create an instance of GameModeManager every time you change the gamemode

Gamemodes

Inside `GameModeManager.cs` there is an Enum located named `Gamemodes`. This Enum is used in conditions to make sure you are in the correct gamemode.

Inventory

Player has an instance of inventory. Inside inventory there is a property called `ItemList`, that is the actual inventory. The rest of the class is full of methods to add or remove an item from the inventory

Menu

The logic inside the menu that makes the buttons send you to either the play section or level editor section

Player

Everything is being explained clearly inside the player class. The player class is one of the biggest classes and one of the more important ones, so make sure you understand everything that's going on in this class.

`float moveSpeed`: the amount of pixels the player moves. One gridtile in the game is 100px so the movement speed is 100.

`Transform movePoint`: center point of the player

`SpriteRenderer spriteRenderer`: player sprite, might be currently unused

`LayerMask stopMovementLayer`: the player cannot pass through objects in this layer

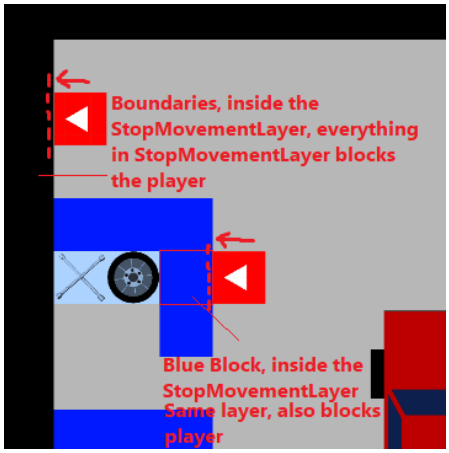


Figure 11: StopMovementLayer in action, every GameObject containing StopMovementLayer will stop the player from moving that direction any further

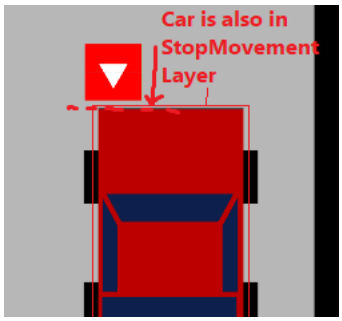


Figure 12: Same for the car, it is also in the StopMovementLayer

`List<CarTask> TaskList`: The tasks the player has to do to complete the level. This tasks are being loaded in from the textfile CarTasks.txt and can be modified in the level editor.

OnCollisionEnter2D

Anytime you see `GameModeManager.Gamemode` inside an if statement, it is to leave out certain unwanted features (line 29).

If Player collides with anything in the play section it will log the item it collided with (see line 32). If the item is collectable, it will destroy the item from the grid, and instantiate a copy of that item in your inventory. All this logic is handled in the inventory class (line 37).

If you collide with the current CarTasks' start-item and you have the required tools, the corresponding minigame will start (line 45).

Movement

Player moves in 4 different directions

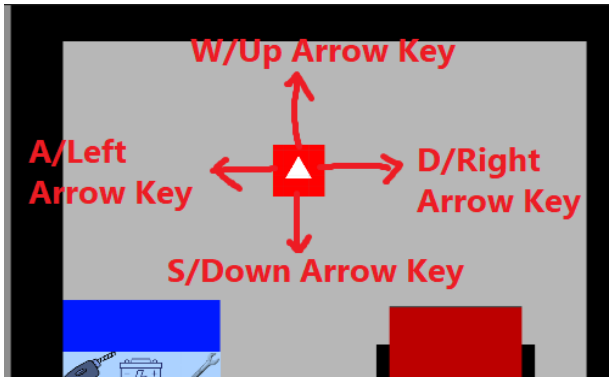


Figure 13: Movement, WASD or Arrow Keys, Move one tile a time

Anytime you see GameModeManager.Gamemode inside an if statement, it is to leave out certain unwanted features (line 113). In this case, the movement only gets triggered if we are in Play mode. That means that if we are in the editor (or any other gamemode), you cannot move the player.

Whenever the player moves into a direction, it will also face the direction it is moving to. This is to make it clear for the player which way they just moved to.

Enable Diagonal Movement

As you can see I have used if, else if, else if, else if for movement (line 115, 124, 132, 140), that means you can only input a single direction at the same time. If you also want to enable diagonal input, you can simply change all else if's to if's. This allows multiple inputs at the same time.

Here's a small demonstration I made of the movement:

Player moves in 4 different directions (can be found back in Movement() inside Player.cs)

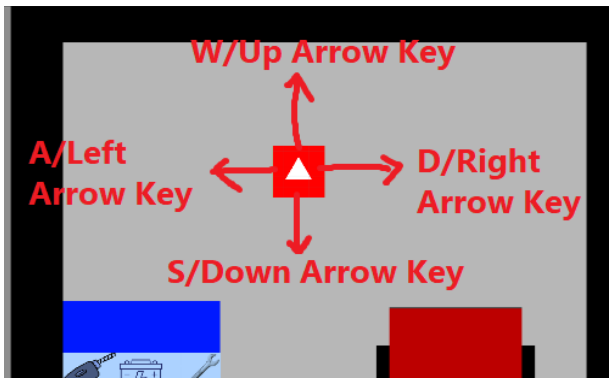


Figure 14: Movement, WASD or Arrow Keys, Move one tile a time

UIItem (& Item)

A tool such as the Wrench has an instance of the class Item. So does every item. But as soon as they appear in the inventory in the play section, we create an UIItem and put the item prefab inside of it. I do this because I want to use the design of the wrench inside the UIItem. The downside is that UIItem and item both have a Box Collider. This means that one time you might click on the UIItem's collider and the other time you might click on the item's collider. Here's what it looks like inside unity

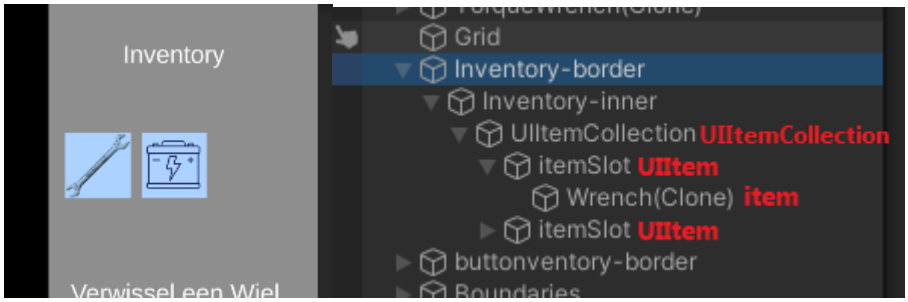


Figure 15: you can see the item being inside a UIItem. This problem can be fixed by disabling the box collider of the item when being put in an UIItem object.

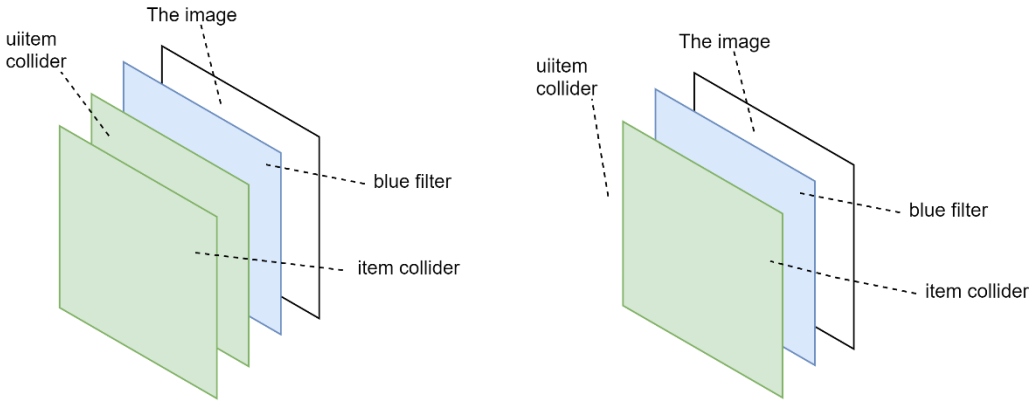


Figure 16: The structure of an UIItem. As you can see the colliders appear on top of one another and it isn't a consistent outcome.

The UIItem is being used inside of CarTasks. For example if you have to remove bolts using a CrossSocketWrench, you first click on the UIItem CrossSocketWrench. Same with the wheel, you have to attach the wheel to the car, so you click on the UIItem wheel.

UIItemCollection

An object that contains all the UIItems.

GameObject SelectedPrefab: The prefab inside the UIItem you selected. (for example, you select Wrench inside the Level editor, SelectedPrefab will be Wrench prefab. If you now click on the grid, it will instantiate a Wrench, because Wrench is the prefab inside SelectedPrefab)

You can click on any item inside the UIItemCollection. When you do so, the UIItem you clicked gains a border around it. this way the user can understand you selected that UIItem.

Buttons

BackToMenuButton

When clicked, brings you back to the menu. Can be triggered in both play section and level editor section.

CarTaskButton

A button that contains a CarTask. When clicked, it will add or remove the corresponding CarTask from CarTasks.txt

If pressed when white, it will turn green and add the CarTask to the data.

If pressed when green, it will turn white and remove the CarTask from the data.

CarTaskButtonCollection

Collection of the CarTaskButtons. Creates a CarTaskButton foreach CarTask.

LoadSaveButton

Loads the save from Grid.txt.

LoadSaveButton also contains a hardcoded prefab list. You need to get rid of this hardcoded list. You can do it by loading in the prefabs from the resources.

LoadTasks

Loads the tasks from the CarTasks.txt

RestartButton

Reloads the play scene.

SaveButton

Currently does nothing. Right now everything gets saved automatically.

CarTasks

CarTasks contains another folder for the CarTasks named Tasks. In the root of CarTasks, you can find the base class and the CarTaskCollection.

Tasks

CarTask1

Child Class of the Base Class CarTask. The first CarTask. This CarTask has been hardcoded mostly.

CarTask2

Child Class of the Base Class CarTask. This CarTask doesn't have a minigame linked to it.

CarTask

The Base Class.

int ID: id of the CarTask, useful for the CarTasks.txt data file.

string Name: name of the CarTask for in the UI.

Items StartItem: the item the player has to interact with to start the CarTask.

string Description: description of the CarTask for in the UI.

Items[] RequiredTools: the tools needed to start the CarTask. This currently gets logged. Would be better to show it in the UI for example where it says placeholder.

Activate(): start the minigame.

Deactivate(): stop the minigame.

CarTaskCollection

Collection of all the CarTasks. If you want to add a new one, you should add it to the constructor.

Level Editor

Eraser

If you click on the eraser, the eraser becomes enabled. If you now click on an item it gets removed.

Grid

If you click on the grid it tries to get SelectedPrefab from UIItemCollection. This could cause an error because you can also just not select an UIItem and click on the grid.

Item

`bool Collectable`: determines if you can collect the item or not. If you collect an item it goes to your inventory.

`Items ItemType`: the type of item it is

`Void OnCollisionEnter2D()`: if item is collectable destroy it. (so we can later make a copy of it and add it to the inventory)

Items

An Enum inside the Item class.

All Currently Existing Items

Prefabs of the Items can be found in either

Assets/Resources/prefabs

Assets/Prefabs

```
public enum Items
{
    None = -1,
    Player = 0,
    BlueBlock = 1,
    GreenBlock = 2,
    PinkBlock = 3,
    Drill = 4, // Boor
    Car = 5, // Auto
    Wheel = 6, // Wiel
    Bolt = 7, // Schroef
    BoltHole = 8, // Gat zonder schroef
    CrossSocketWrench = 9, // Kruissleutel
    TorqueWrench = 10, // Moment Sleutel
    EngineHood = 11, // Motor Kap
    CarBattery = 12, // Accu
    Wrench = 13, // Steek Sleutel
}
```

Scenes

LevelEditorScene

Sets the gamemode to Level Editor when loaded

MenuScene

Sets the gamemode to Menu when loaded

PlayScene

Sets the gamemode to Play when loaded

Prefabs

Some are loaded and accessed during runtime. If you want to access a Prefab during runtime make sure you place it in Resources/prefabs/. During runtime you can access them with `Resources.Load("prefabs/ThePrefabYouWantToLoadIn.")`.

Inside LoadSaveButton there still is a hardcoded PrefabList. Because I didn't know yet how to access prefabs during runtime. It would be good to remove that hardcoded PrefabList and access them from runtime instead.

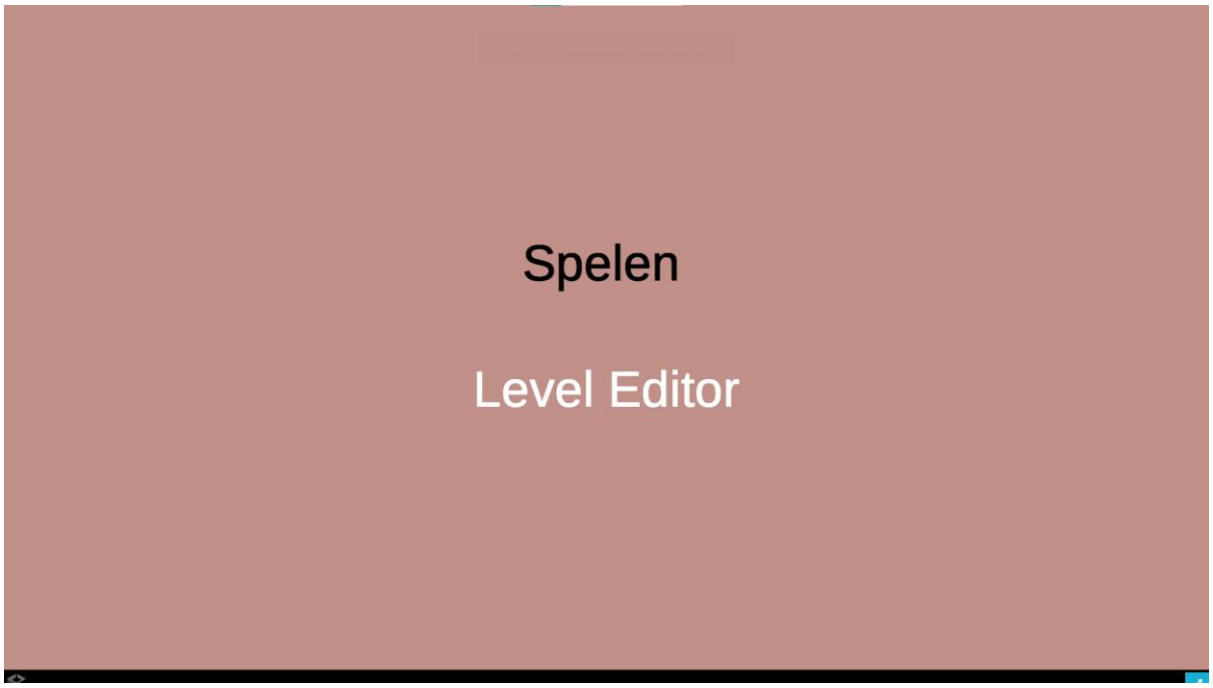
Common Actions

Create a level

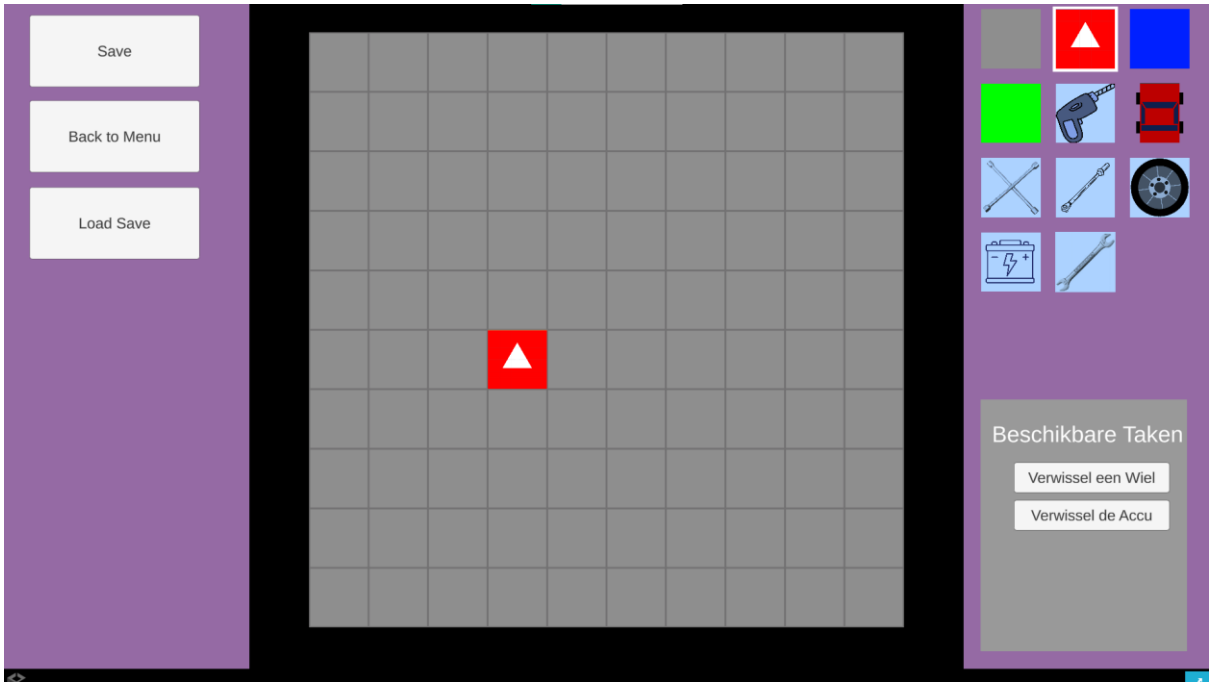
For creating a level there are a few steps.

Disclaimer: If you click on a CarTaskButton on the current Itch.io build the save becomes corrupt.

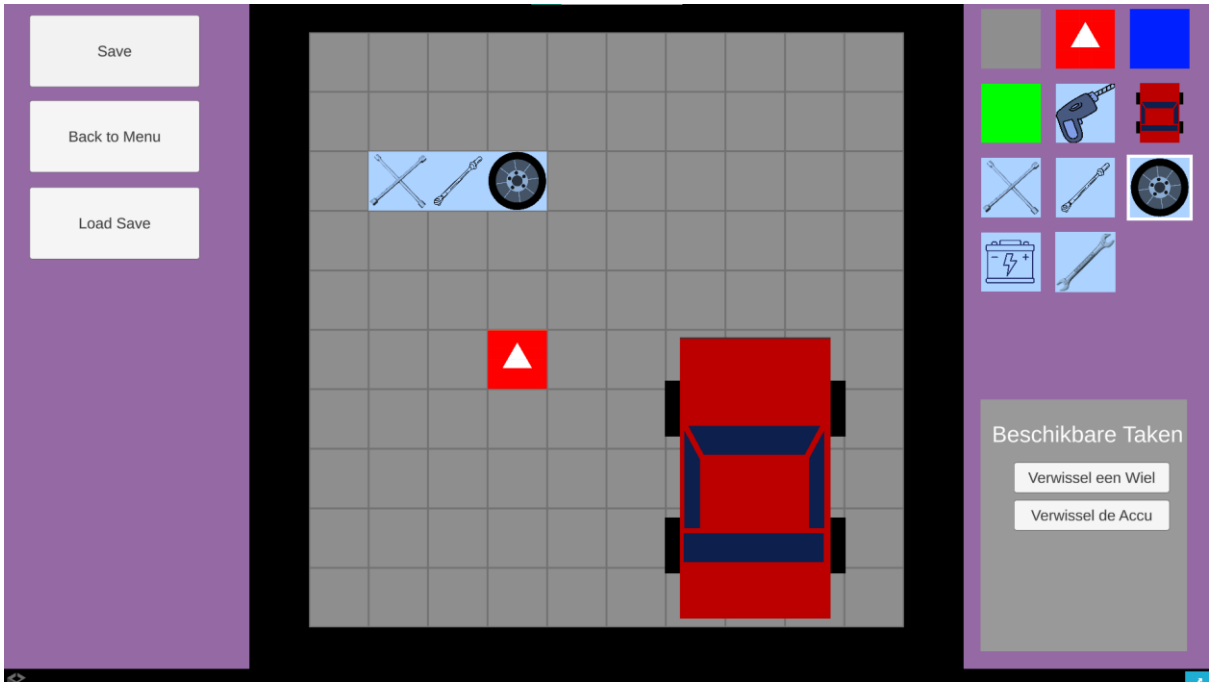
1. Navigate to the Level Editor by clicking on the text "Level Editor".



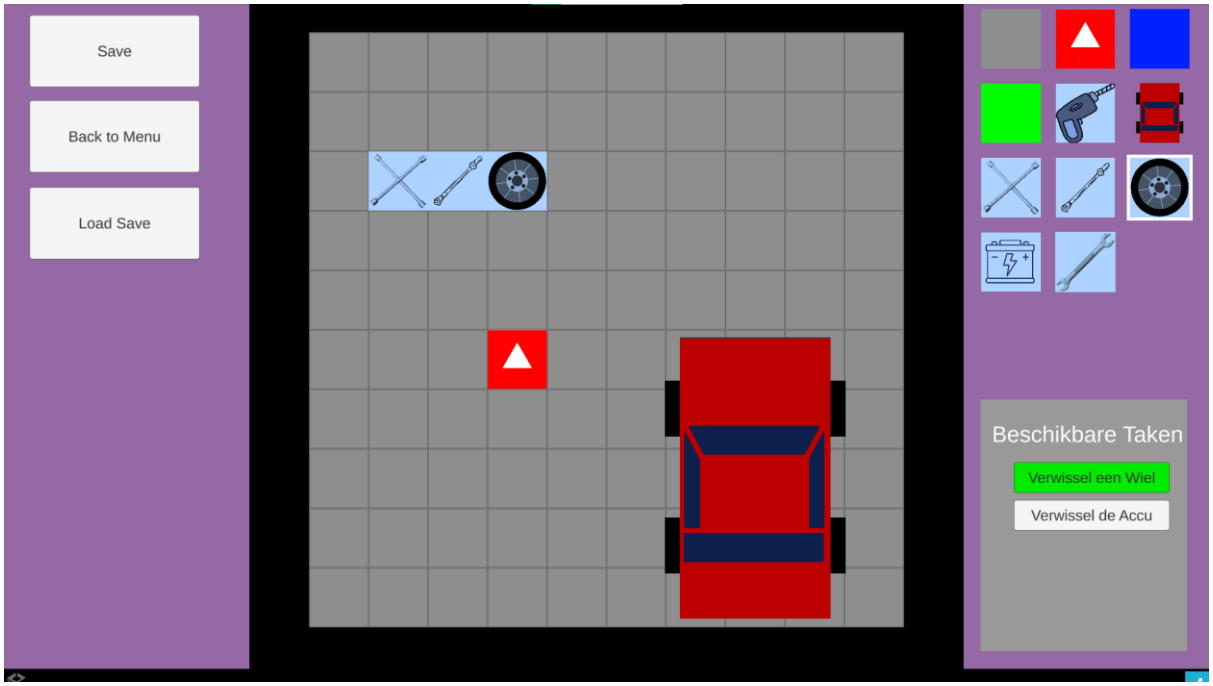
2. Select the Player and click somewhere on the grid to add a Player.



3. Do the same for a few tools. I chose to add the Cross Socket Wrench, Torque Wrench and a Tire. I also added a Car. These are the minimum ingredients to Start CarTask1 – Change a Tire.



4. Next up, in available tasks we click on “Verwissel een Wiel” which stands for Change a Tire. Now we assigned that certain CarTask to the player.

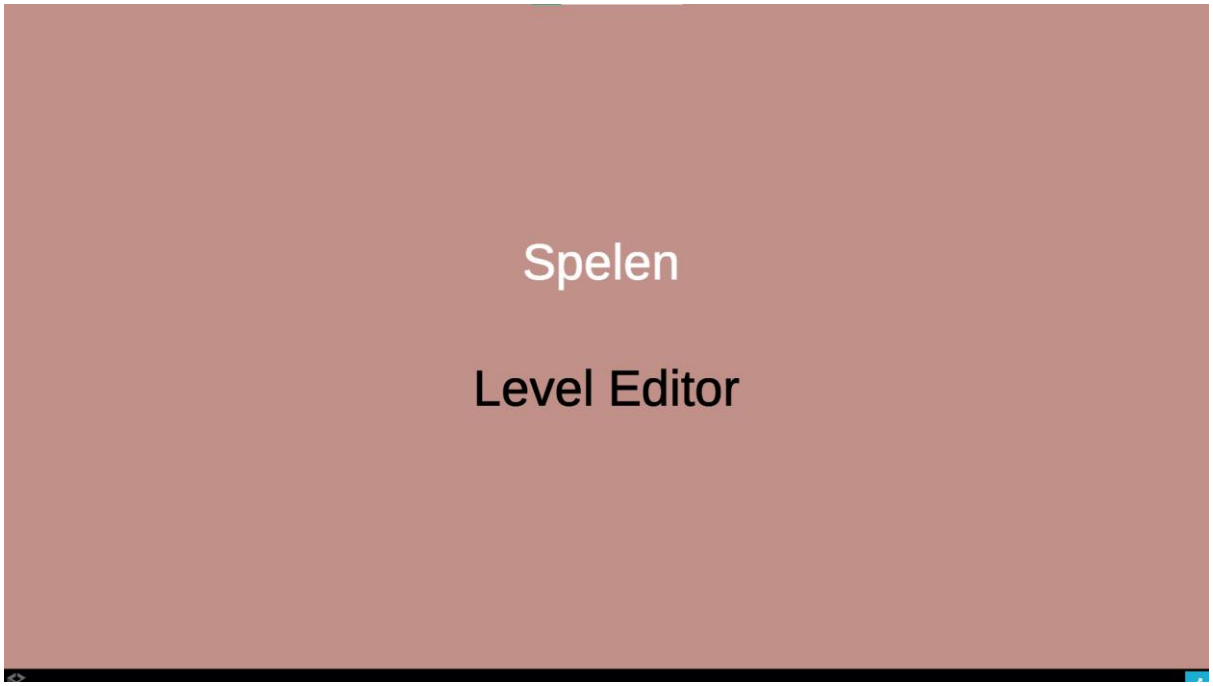


5. Our work in the editor is done. All our actions have been saved automatically. The objects on the grid have been saved to Grid.txt and the assigned tasks have been assigned to CarTasks.txt. Now we can navigate back to the menu using the “Back to Menu” button.

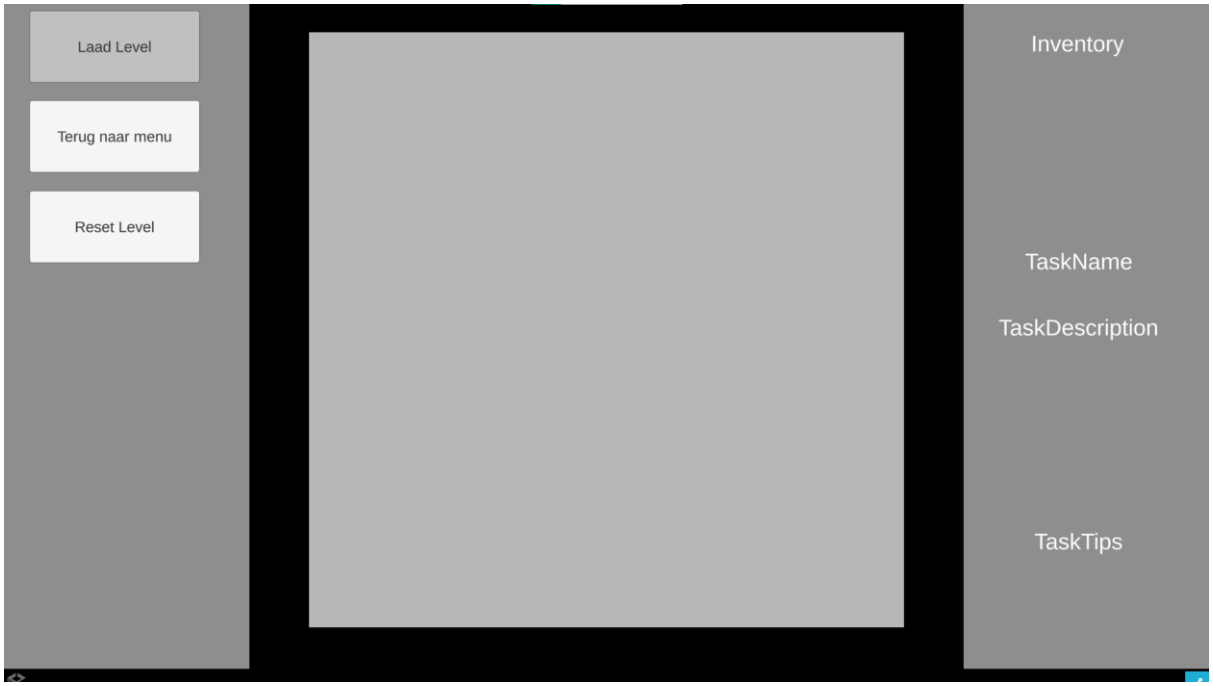
Play a Level

To play a level you must have a Level created before

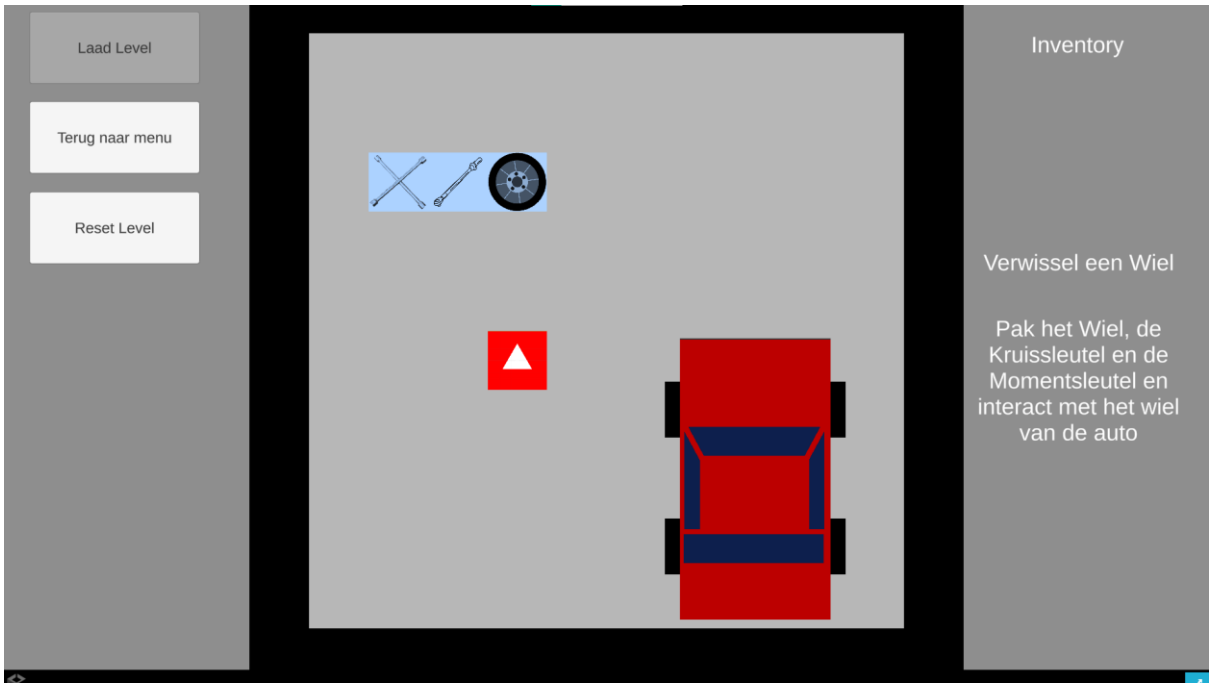
1. Navigate to the play section by clicking on the text “Spelen”.



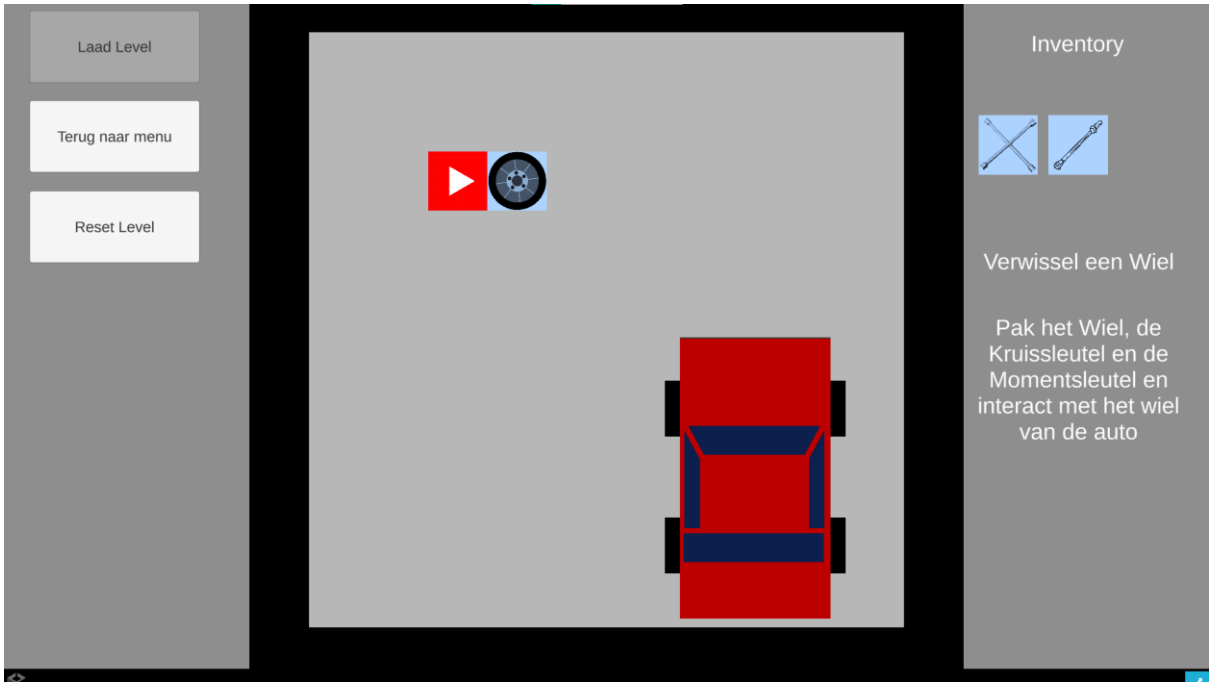
2. Load in the level you just created by clicking on “Laad Level”.



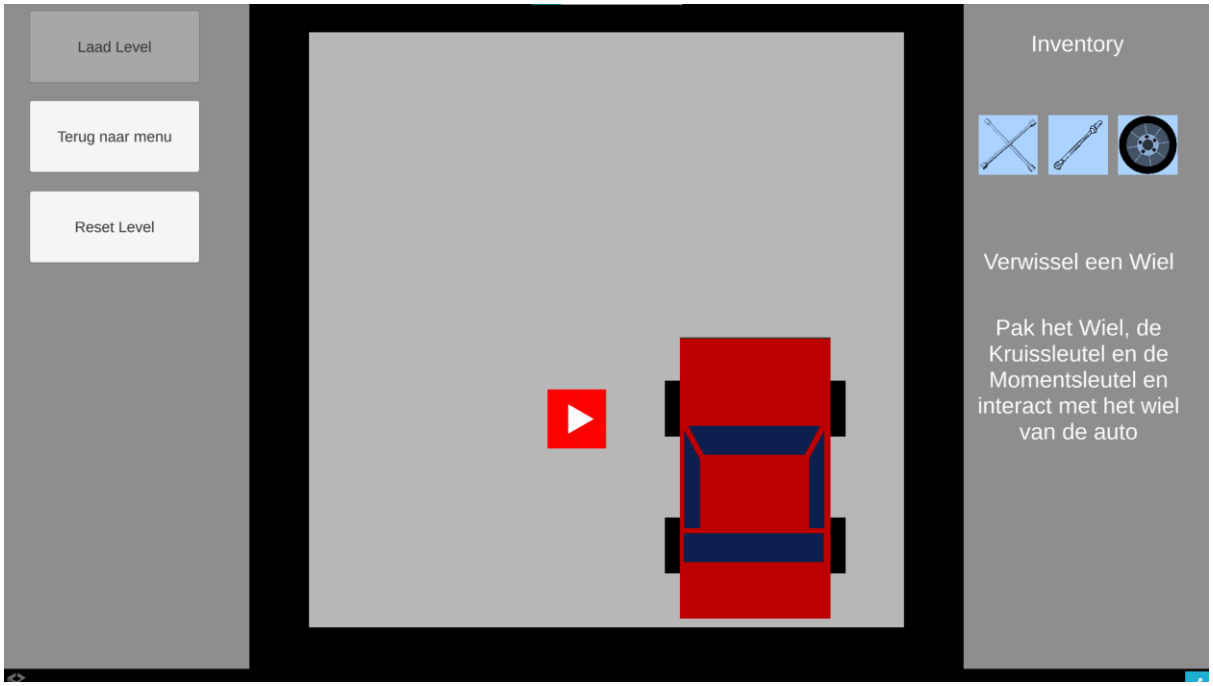
3. Now that you have loaded in the level you can start playing the game. On the right you have a short description of what you should do. It tells you to pick up the Tire, Cross Socket Wrench, Torque wrench and next interact with the tire of the car.



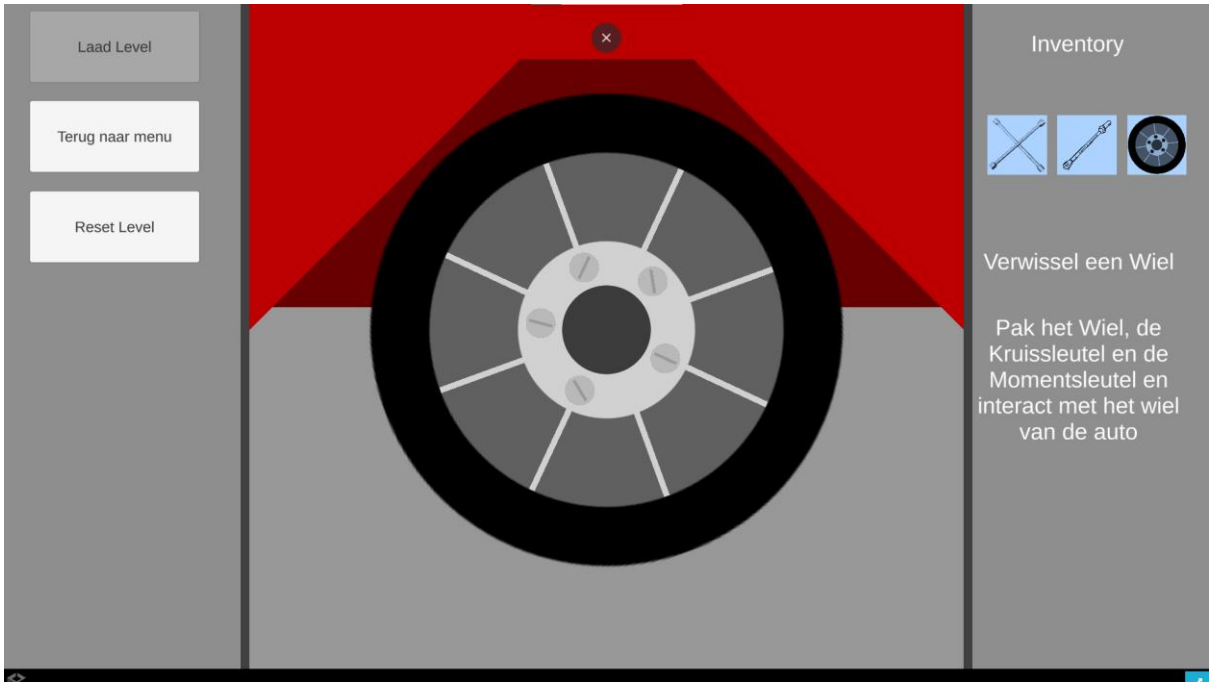
4. Pickup the Items.



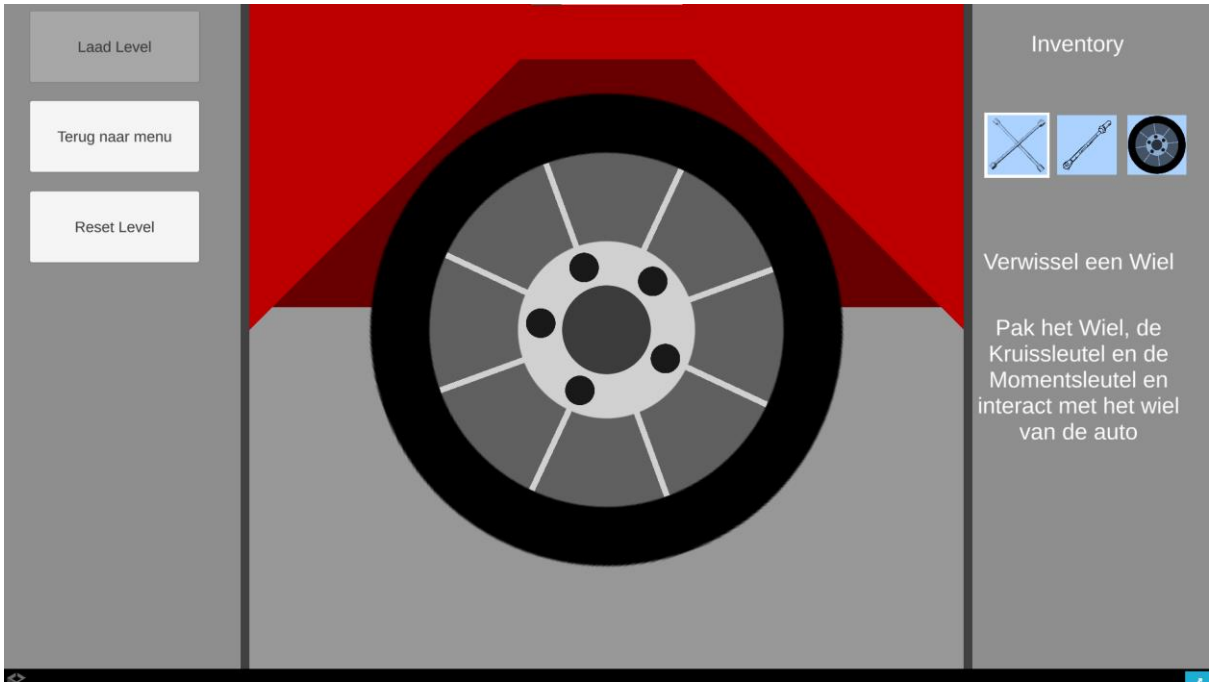
5. After picking up all items you can move towards the tire of the car



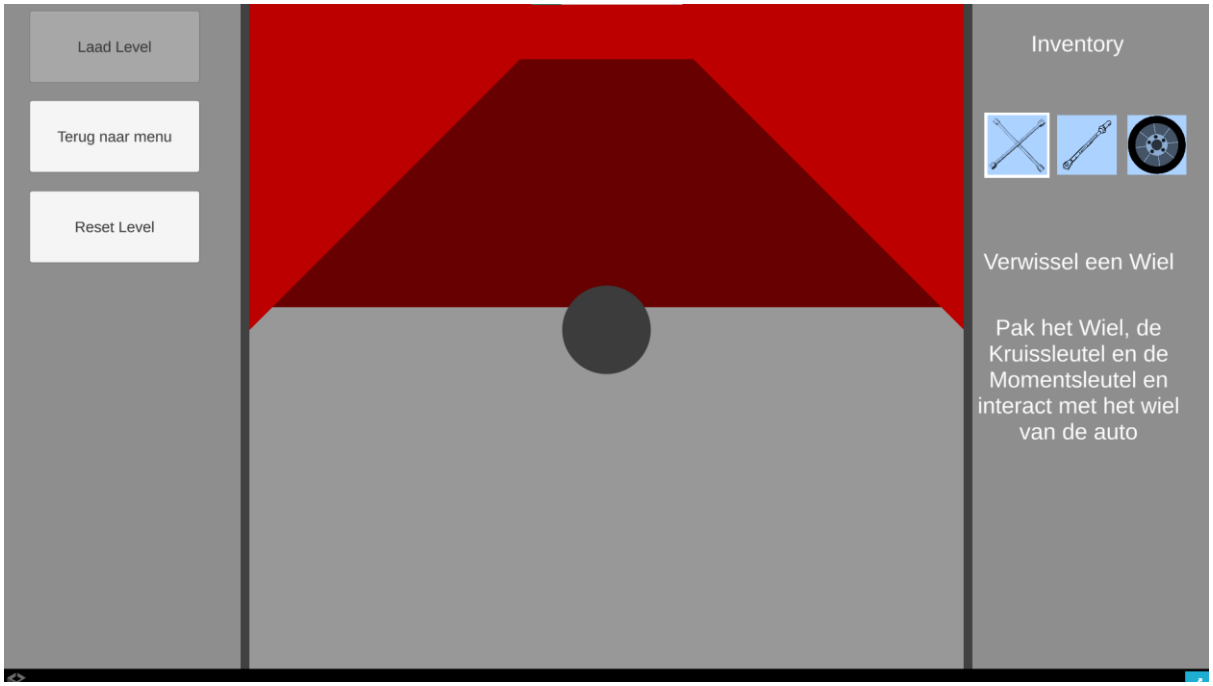
6. When interacted with the tire of the car you are confronted with this CarTask screen. Now your task is to change the tire with the one you collected.



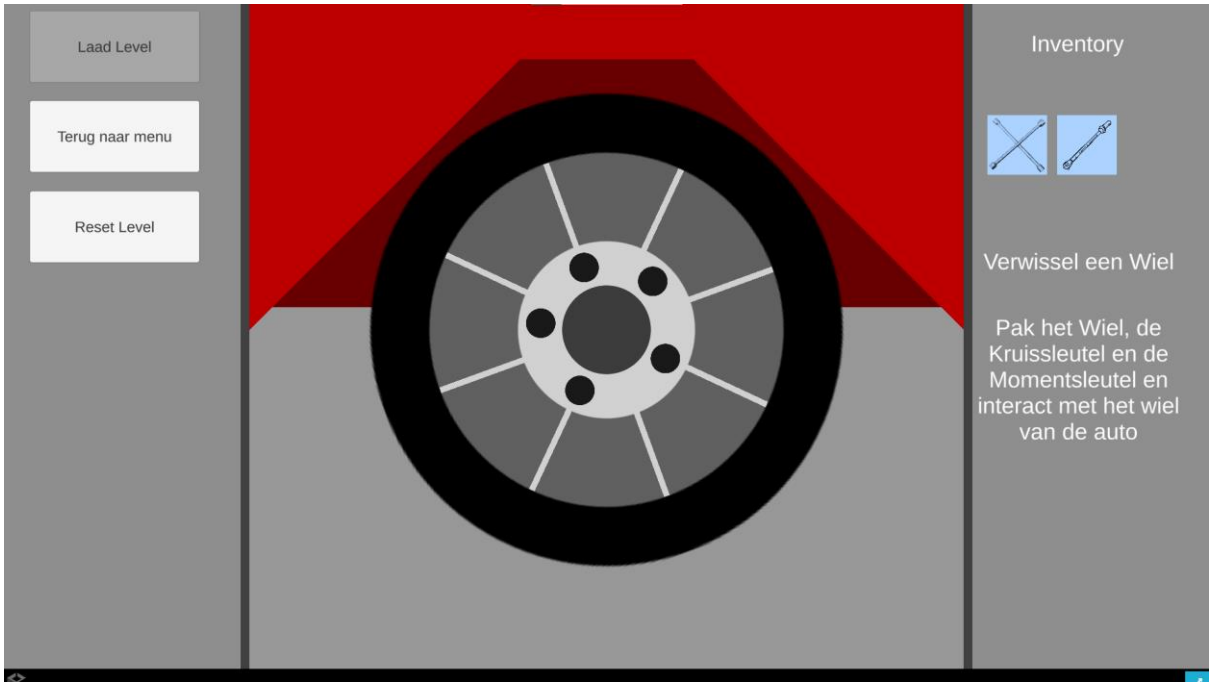
7. Select Cross Socket Wrench and click on the bolts to remove them.



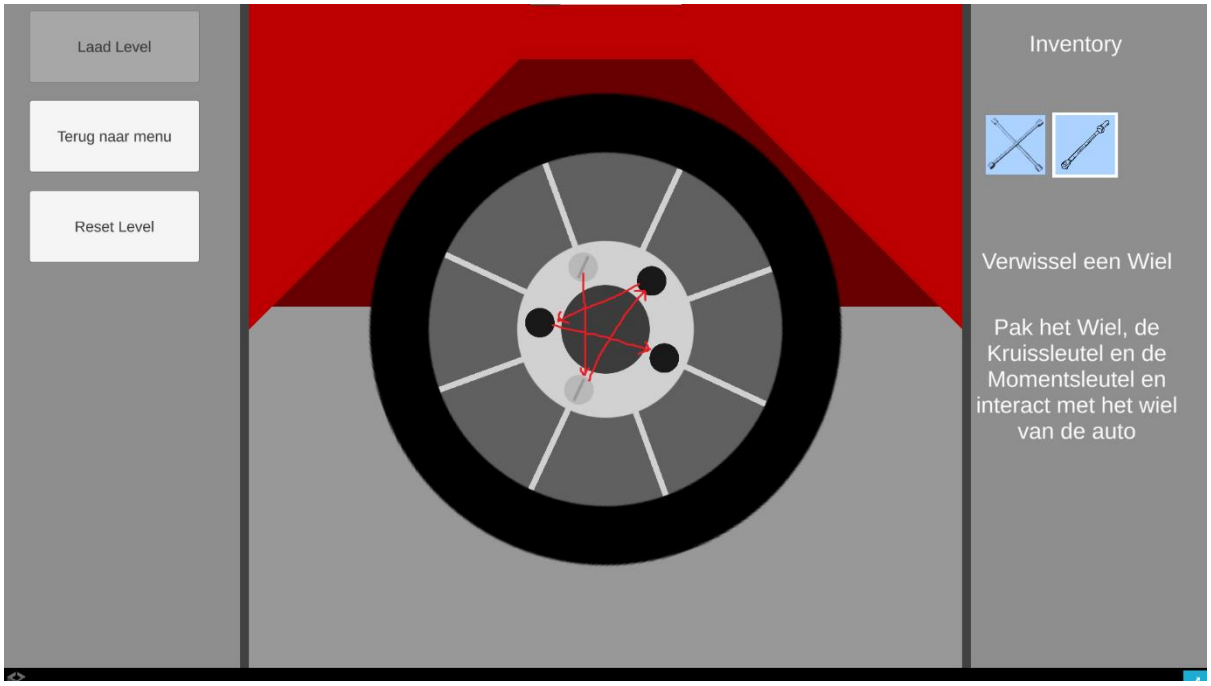
8. Click on the Wheel.



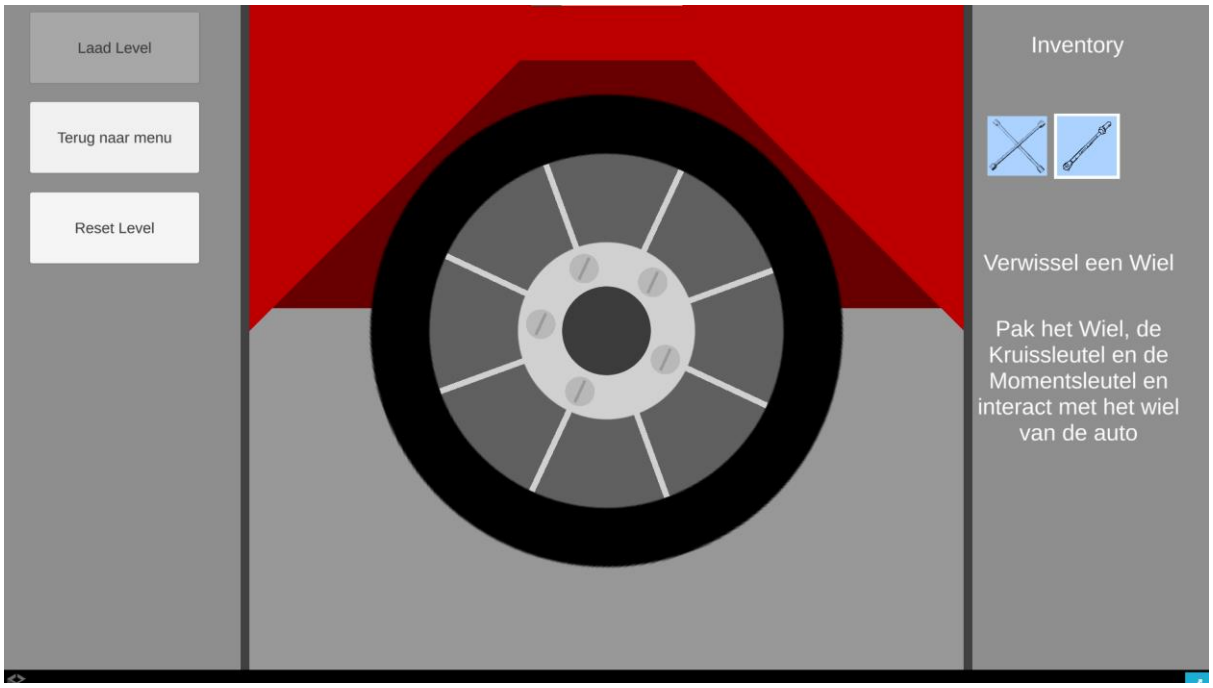
9. Click on the Wheel in your inventory.



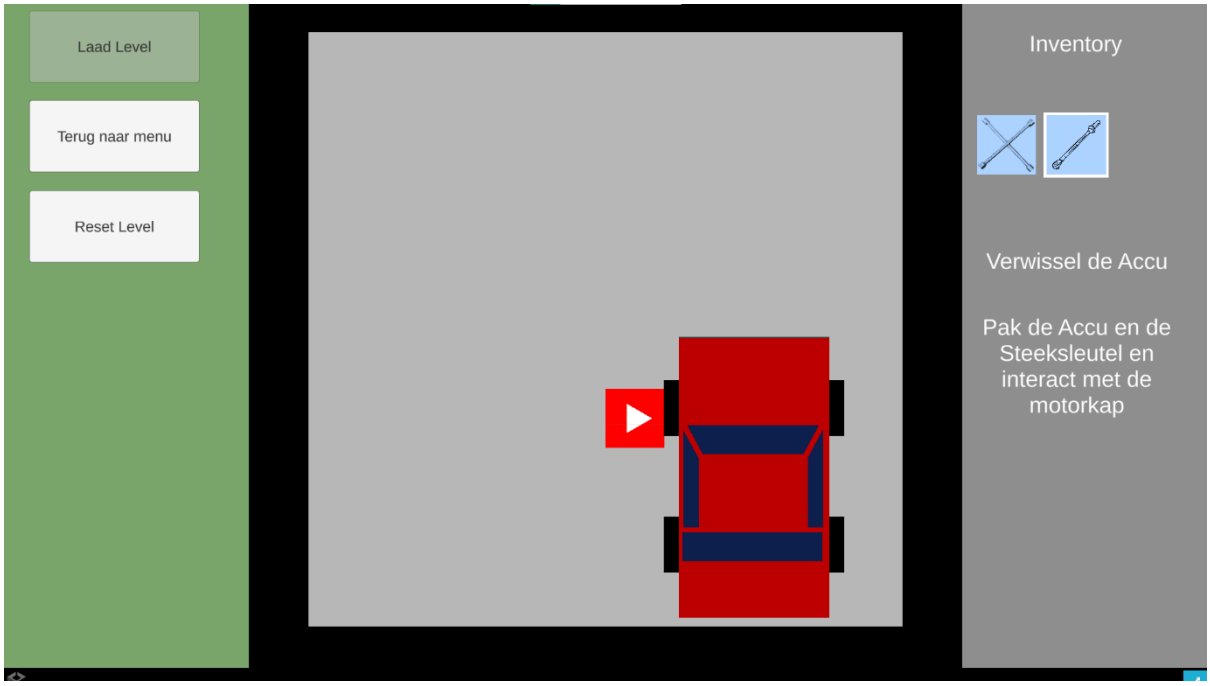
10. Apply the bolts crosswise with the torquewrench.



11. Click on the wheel to complete the task



12. Task completed, if you are on the current unity build it gets an error because it tried to load the next task but we only selected one. If you are on the itch build it will load in the next task since the tasks on itch are hardcoded to give you all existing tasks.



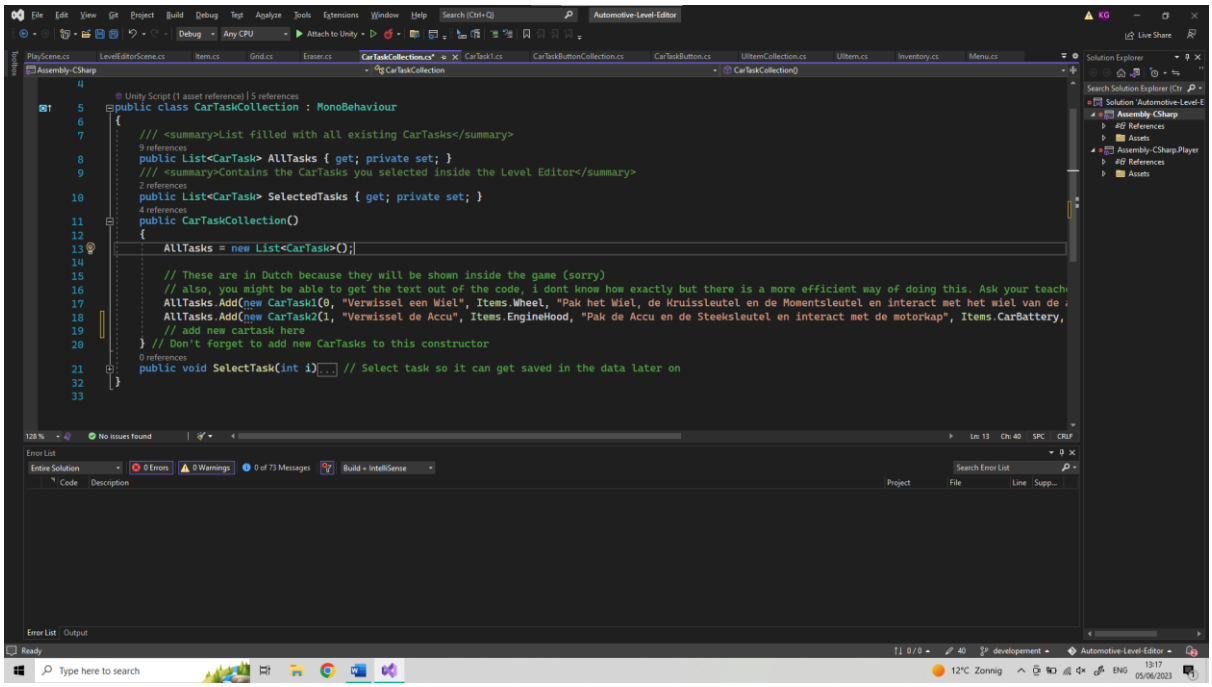
Add a new CarTask

Quite rapid and easy process.

1. Make a new class and make it inherit from the CarTask abstract class

2. Go to CarTaskCollection.cs

3. Create an instance of the newly created CarTask to the constructor (see line 19 in the image below)



```
4
5   public class CarTaskCollection : MonoBehaviour
6   {
7       /// <summary>List filled with all existing CarTasks</summary>
8       public List<CarTask> AllTasks { get; private set; }
9       /// <summary>Contains the CarTasks you selected inside the Level Editor</summary>
10      public List<CarTask> SelectedTasks { get; private set; }
11      public CarTaskCollection()
12      {
13          AllTasks = new List<CarTask>();
14
15          // These are in Dutch because they will be shown inside the game (sorry)
16          // also, you might be able to get the text out of the code, i dont know how exactly but there is a more efficient way of doing this. Ask your teach
17          AllTasks.Add(new CarTask1(0, "Verwissel een Wiel", Items.Wheel, "Pak het Wiel, de Kruissleutel en de Momentsleutel en interact met het wiel van de
18          AllTasks.Add(new CarTask2(1, "Verwissel de Accu", Items.EngineHood, "Pak de Accu en de Steeksleutel en interact met de motorkap", Items.CarBattery,
19          // add new cartask here
20      } // Don't forget to add new CarTasks to this constructor
21      public void SelectTask(int i)... // Select task so it can get saved in the data later on
32  }
33
```

That's it. Dynamically there will be created a CarTaskButton inside the Level Editor. You can assign this task to the player by clicking that button and it will work as any other task in the play section. It just doesn't have a minigame linked to it.

Link a Minigame to the new CarTask

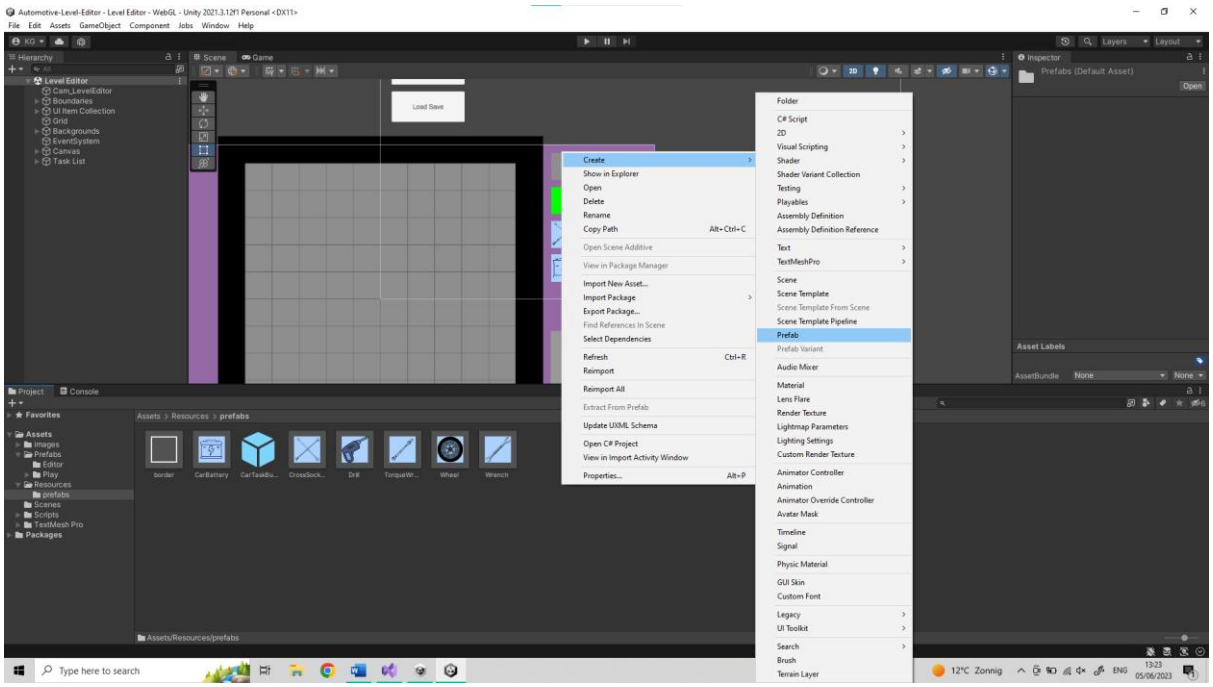
To link a minigame to a CarTask you can make use of the Activate and Deactivate methods. You can look inside CarTask1 for examples. There I used Activate to move the cam to the minigame.

The minigame itself you will have to make yourself. I decided to quickly hardcode it to just have something. I'd advise not hardcoding the minigames so you can reuse some of the components like removing bolts for example.

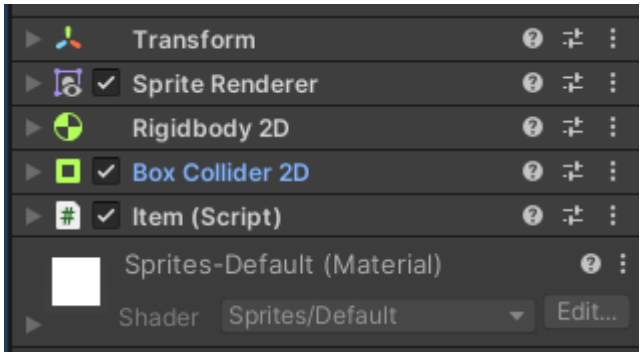
Add a new Item

This contains lots of old low level code so this process is quite lengthy. **Instead of following these instructions I would advise removing the hardcoding inside this process. It will make the process shorter and more maintainable.**

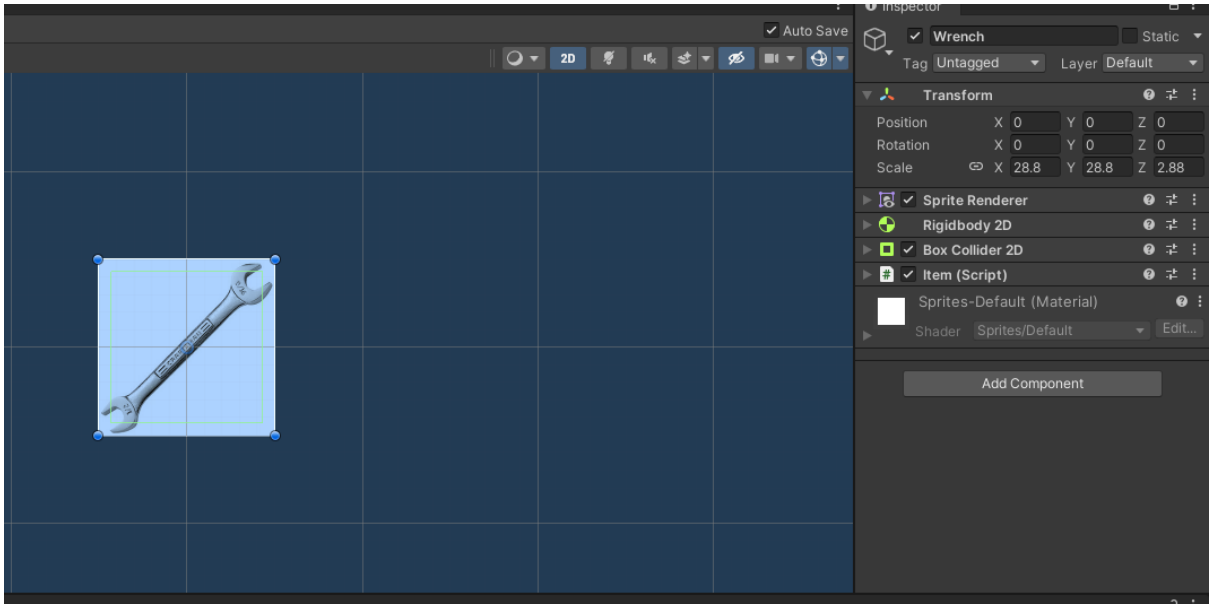
1. Go to resources and create a new Prefab in the prefabs folder.



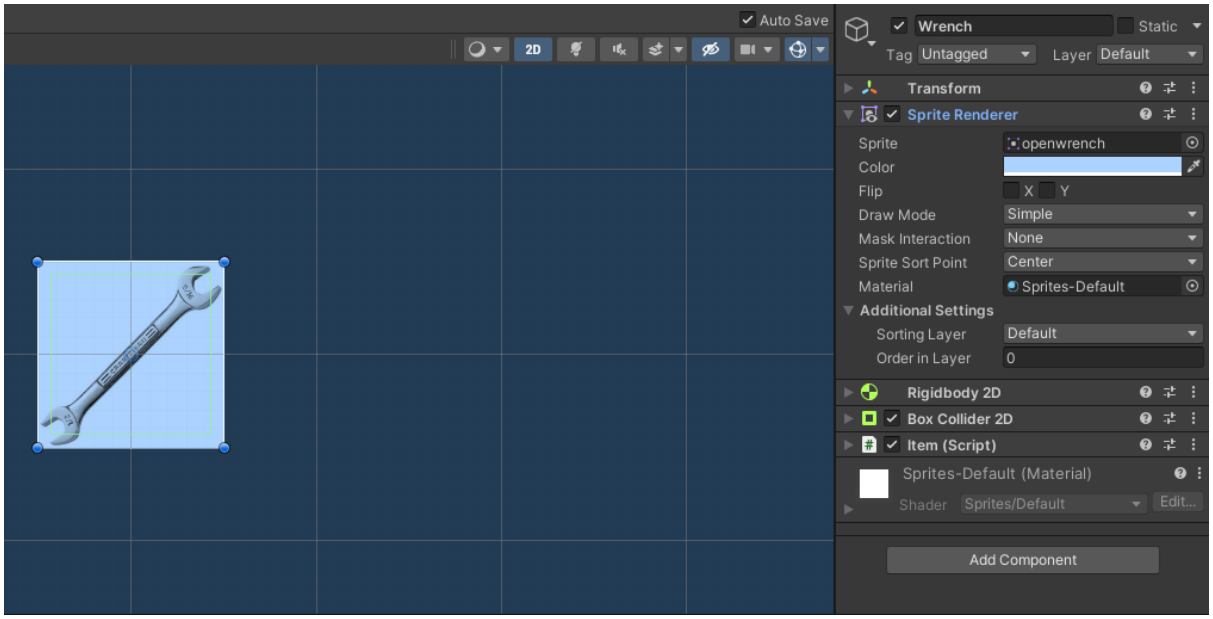
2. Give it these components



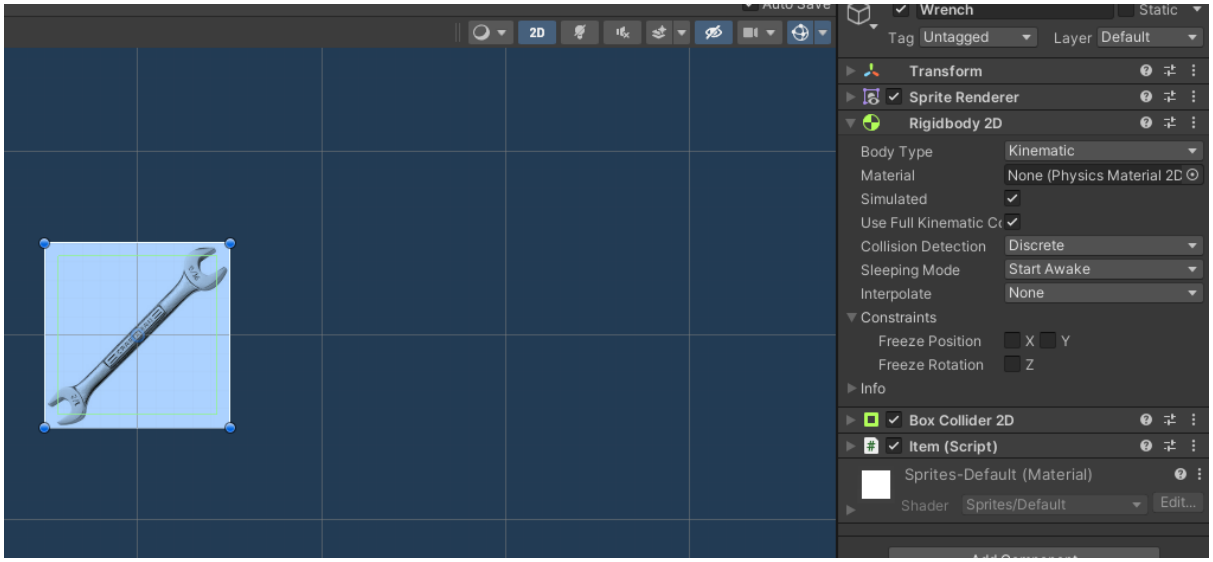
3. Set the Transform Right. I used Scale 28.8, but you want it to get the same size as you see left.



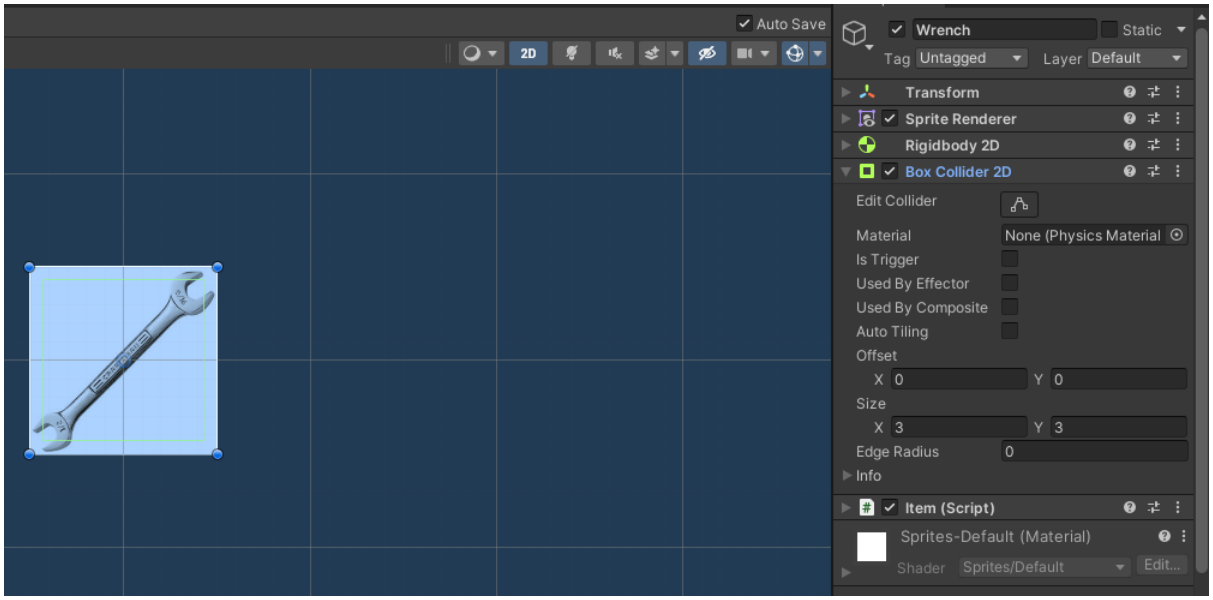
4. Inside the sprite renderer choose an image of your choice. If it is a tool, make sure to make it blue like all the other tools.



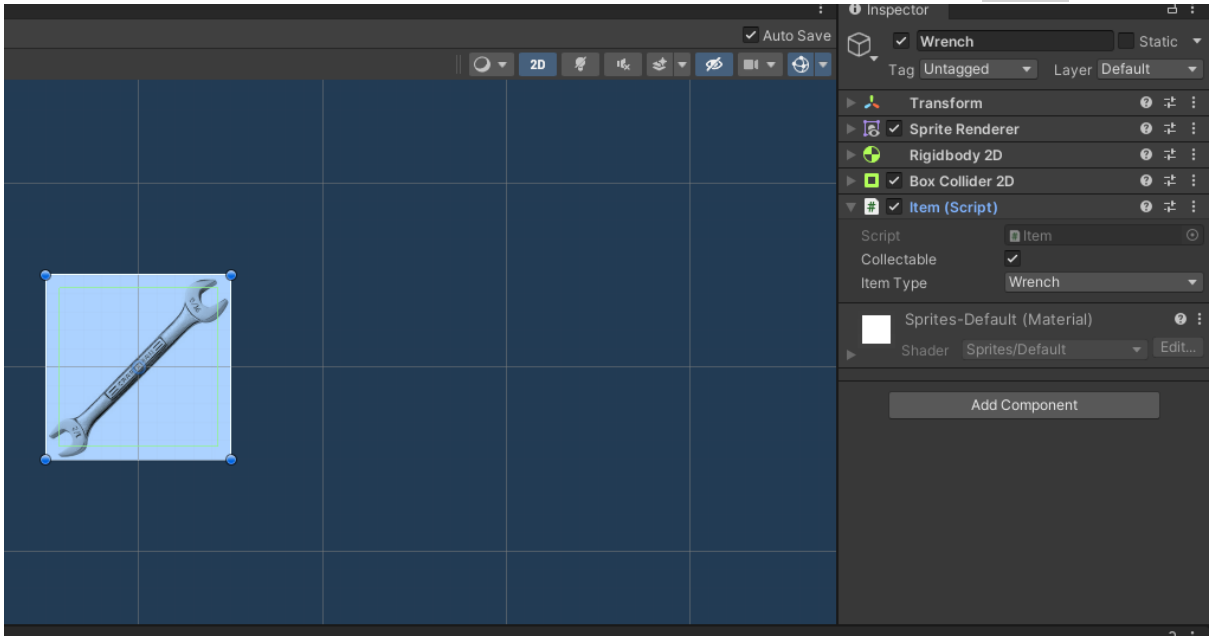
5. **Make sure you have added Rigidbody2D not RigidBody.** Change BodyType to kinematic, check Simulated and Use Full Kinematic, collision detection on discrete.



6. **Make sure you add BoxCollider2D, not BoxCollider. Also make sure the green collision box is a bit smaller than the tool itself**.



7. Add the Item script. If it is collectable, check the collectable box. Inside Item Type choose the Item you are adding. If it is not present you will have to add it to the Items enum inside Item.cs

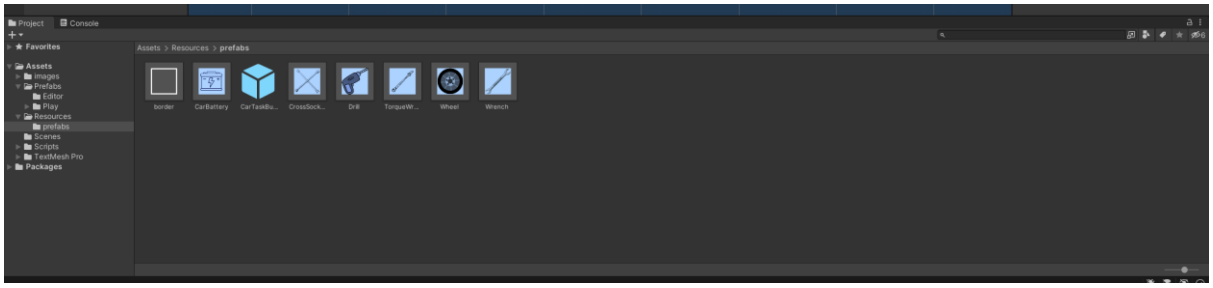


8. Add your item to the enum if you didn't already. **Do not change the numbers that are already assigned, it will mess with data inside the app.**

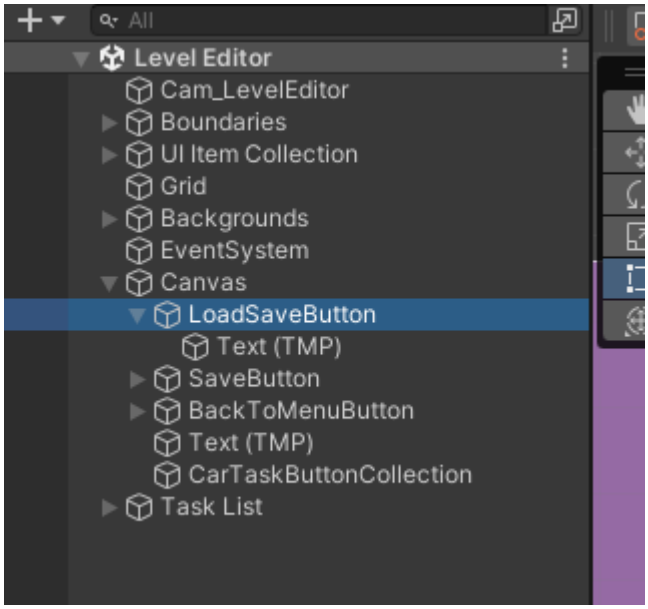
```
    // I've put the Dutch names of tools next to it
public enum Items
{
    None = -1,
    Player = 0,
    BlueBlock = 1,
    GreenBlock = 2,
    PinkBlock = 3,
    Drill = 4, // Boor
    Car = 5, // Auto
    Wheel = 6, // Wiel
    Bolt = 7, // Schroef
    BoltHole = 8, // Gat zonder schroef
    CrossSocketWrench = 9, // Kruissleutel
    TorqueWrench = 10, // Moment Sleutel
    EngineHood = 11, // Motor Kap
    CarBattery = 12, // Accu
    Wrench = 13, // Steek Sleutel
}

public class Item : MonoBehaviour
{
```

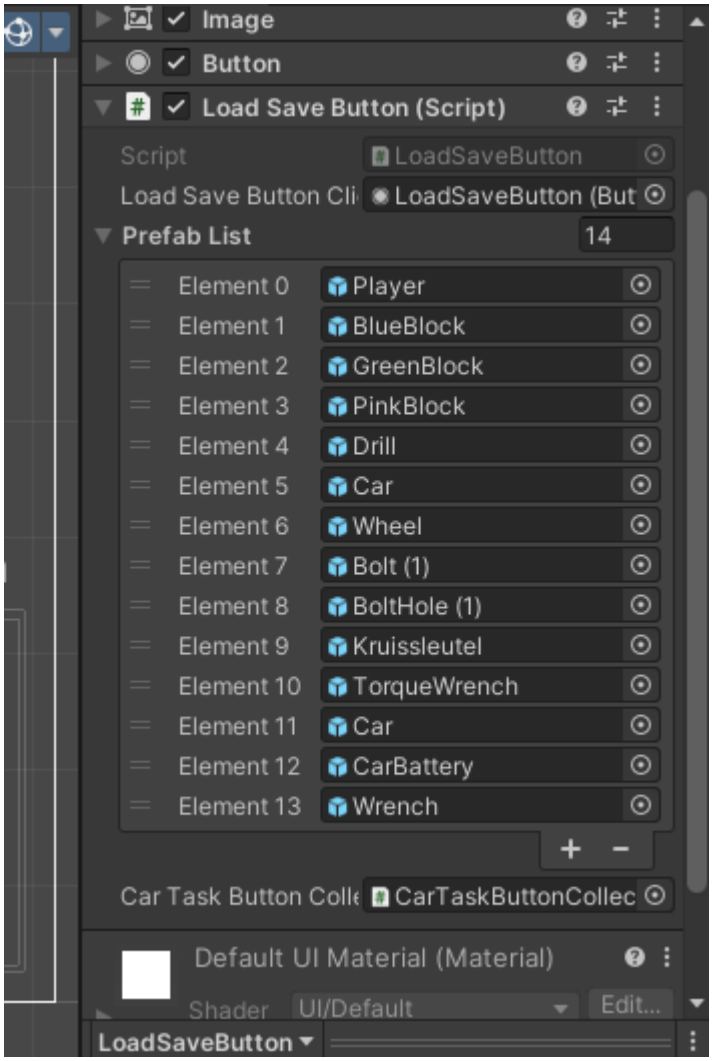
9. Now you should have your prefab added here. As I mentioned earlier, it's low level code so we need to follow a few more steps.



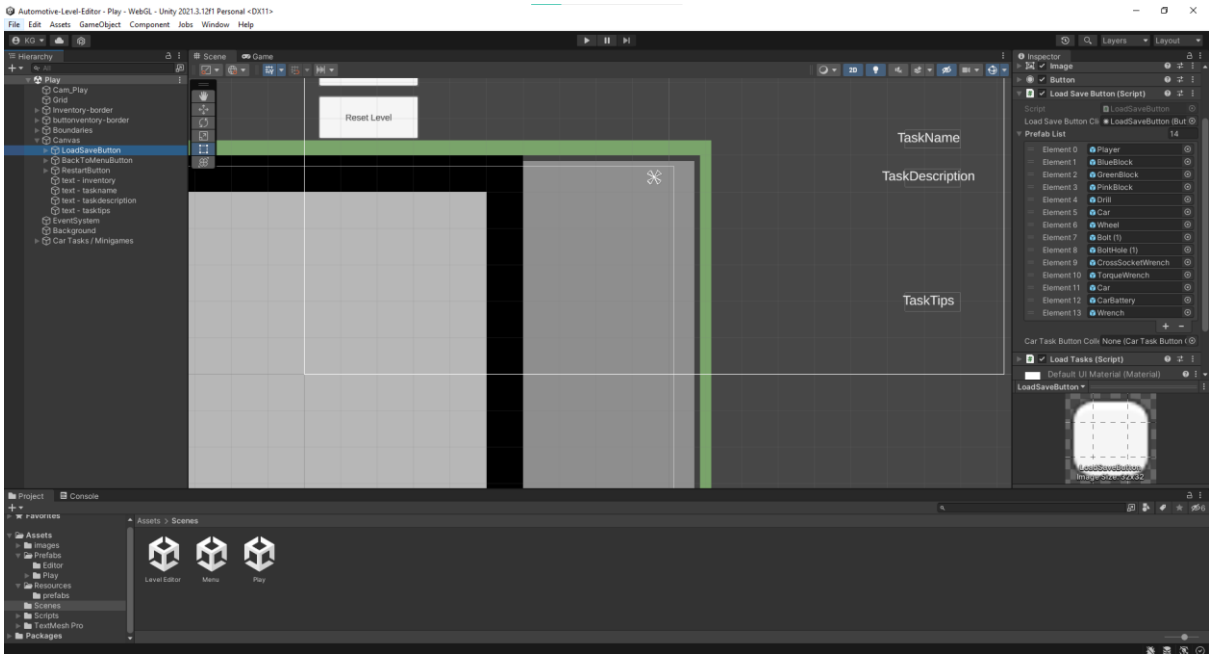
10. Go to the Level Editor Scene and click on LoadSaveButton inside the Canvas.



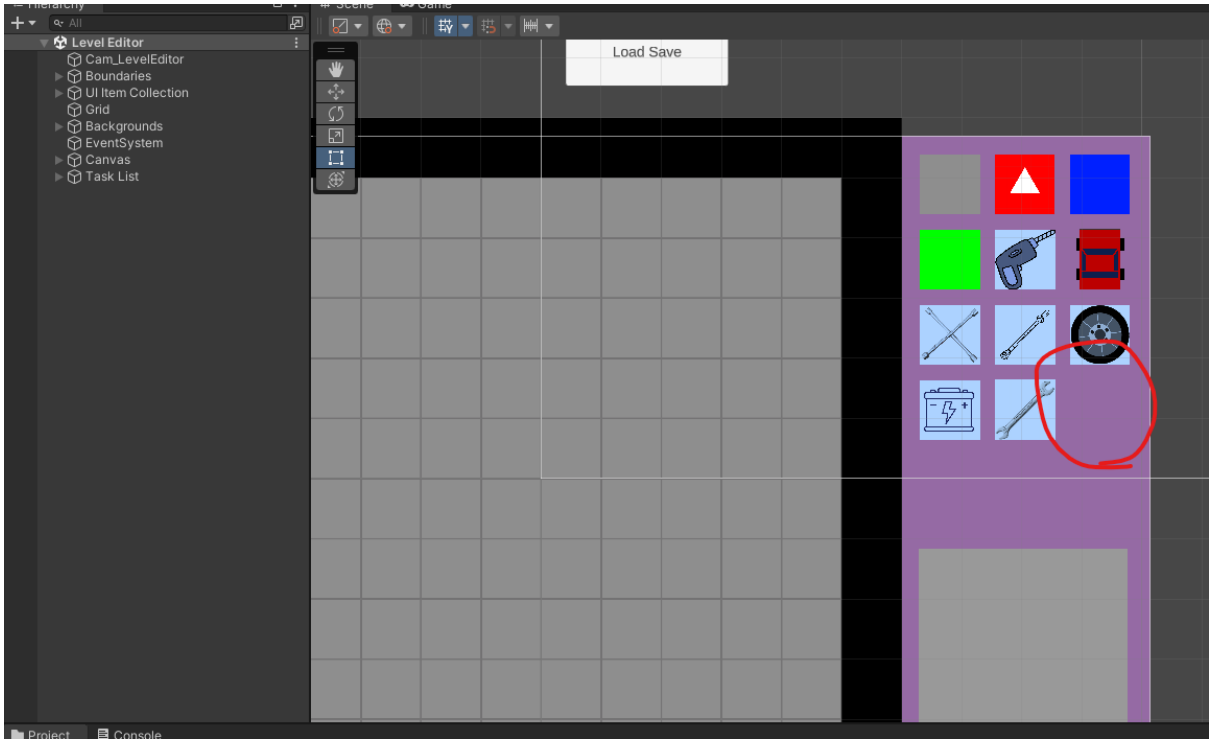
11. Navigate to the LoadSaveButton script. Add your newly made Prefab to the prefablist. **Do not mess with the other elements, also do not rename the PrefabList variable or change it to private it will delete all values.**



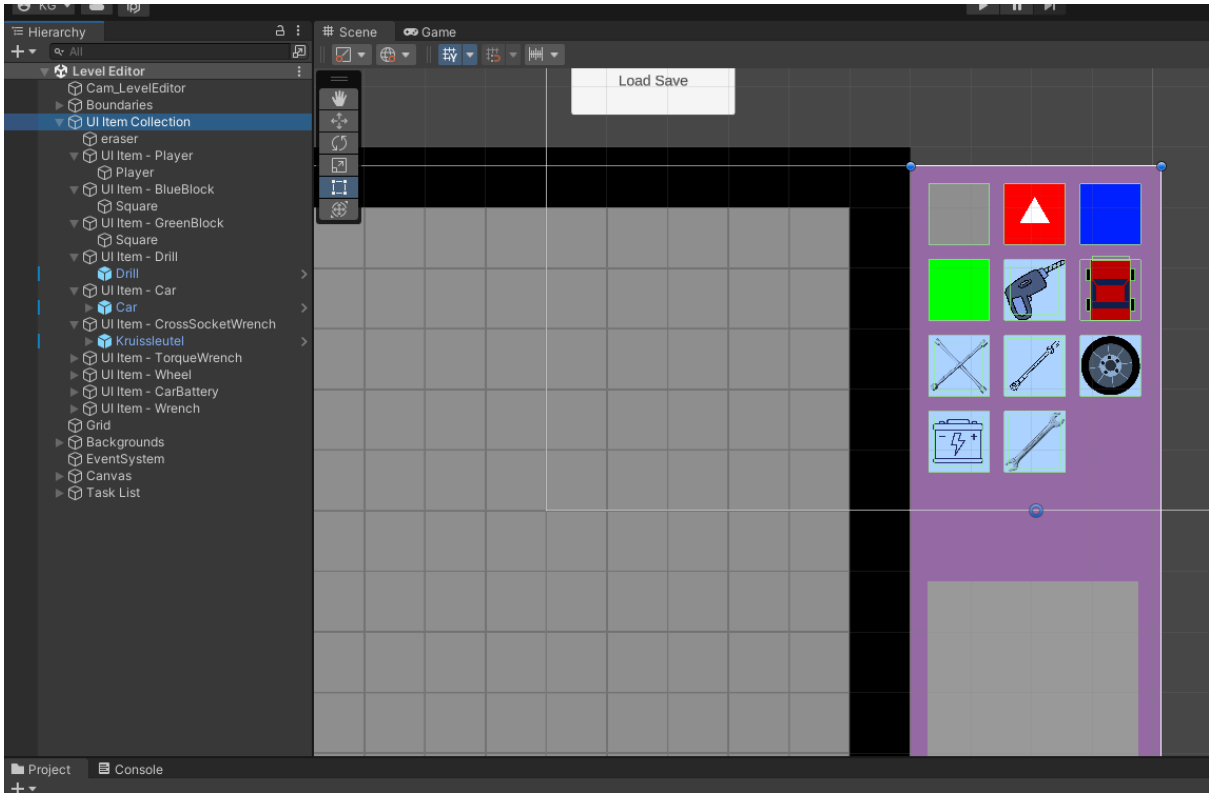
12. Do the exact same thing in the play scene. This is necessary because currently for loading in objects from the save we use the prefablist to find prefabs. You can easily replace this with Resources.Load()



13. As last we also have to manually add it to the level editor UIItemCollection. First open de Level Editor Scene



14. You can Copy paste a UI Item and as child add your object to show it in the UIItemCollection.



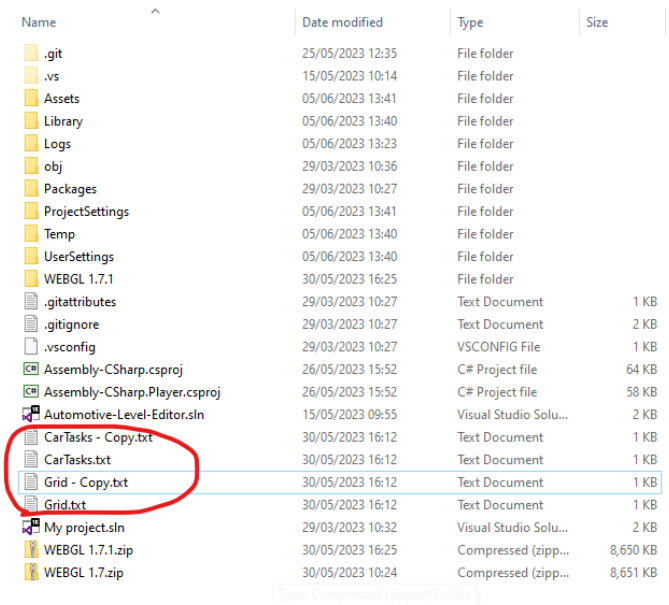
Make a backup of a level

If you have made a level you really like, you can back it up like the following

1. locate the 2 data files.

Name	Date modified	Type	Size
.git	25/05/2023 12:35	File folder	
.vs	15/05/2023 10:14	File folder	
Assets	05/06/2023 13:41	File folder	
Library	05/06/2023 13:40	File folder	
Logs	05/06/2023 13:23	File folder	
obj	29/03/2023 10:36	File folder	
Packages	29/03/2023 10:27	File folder	
ProjectSettings	05/06/2023 13:41	File folder	
Temp	05/06/2023 13:40	File folder	
UserSettings	05/06/2023 13:40	File folder	
WEBGL 1.7.1	30/05/2023 16:25	File folder	
.gitattributes	29/03/2023 10:27	Text Document	1 KB
.gitignore	29/03/2023 10:27	Text Document	2 KB
.vsconfig	29/03/2023 10:27	VSCONFIG File	1 KB
Assembly-CSharp.csproj	26/05/2023 15:52	C# Project file	64 KB
Assembly-CSharp.Player.csproj	26/05/2023 15:52	C# Project file	58 KB
Automotive-Level-Editor.sln	15/05/2023 09:55	Visual Studio Solu...	2 KB
CarTasks.txt	30/05/2023 16:12	Text Document	1 KB
Grid.txt	30/05/2023 16:12	Text Document	1 KB
My project.sln	29/03/2023 10:32	Visual Studio Solu...	2 KB
WEBGL 1.7.1.zip	30/05/2023 16:25	Compressed (zipp...	8,650 KB
WEBGL 1.7.zip	30/05/2023 10:24	Compressed (zipp...	8,651 KB

2. Make Copies of them.



Name	Date modified	Type	Size
.git	25/05/2023 12:35	File folder	
.vs	15/05/2023 10:14	File folder	
Assets	05/06/2023 13:41	File folder	
Library	05/06/2023 13:40	File folder	
Logs	05/06/2023 13:23	File folder	
obj	29/03/2023 10:36	File folder	
Packages	29/03/2023 10:27	File folder	
ProjectSettings	05/06/2023 13:41	File folder	
Temp	05/06/2023 13:40	File folder	
UserSettings	05/06/2023 13:40	File folder	
WEBGL 1.7.1	30/05/2023 16:25	File folder	
.gitattributes	29/03/2023 10:27	Text Document	1 KB
.gitignore	29/03/2023 10:27	Text Document	2 KB
.vsconfig	29/03/2023 10:27	VSCONFIG File	1 KB
Assembly-CSharp.csproj	26/05/2023 15:52	C# Project file	64 KB
Assembly-CSharp.Player.csproj	26/05/2023 15:52	C# Project file	58 KB
Automotive-Level-Editor.sln	15/05/2023 09:55	Visual Studio Solu...	2 KB
CarTasks - Copy.txt	30/05/2023 16:12	Text Document	1 KB
CarTasks.txt	30/05/2023 16:12	Text Document	1 KB
Grid - Copy.txt	30/05/2023 16:12	Text Document	1 KB
Grid.txt	30/05/2023 16:12	Text Document	1 KB
My project.sln	29/03/2023 10:32	Visual Studio Solu...	2 KB
WEBGL 1.7.1.zip	30/05/2023 16:25	Compressed (zipp...	8,650 KB
WEBGL 1.7.zip	30/05/2023 10:24	Compressed (zipp...	8,651 KB

Whenever you want to use the save file you backed up you simply change the name of the copied file to the original. So you would change `Grid – Copy.txt` to `Grid.txt` to use it. To simplify this process and make it more user friendly you can research how to make use of OpenFileDialog and SaveFileDialog.

Room for improvement

Bugs

I made use of a GitHub project board to keep track of the bugs. On this board you can see all current bugs, and code smells. The project board is public to everyone you just can't edit it without permission. If you want to take over the board for your semester let me know (Discord: CrossyChainsaw#1706).

Online Public Project Board: https://github.com/users/CrossyChainsaw/projects/2/views/1

Recommendations

Most of this is faulty code that does work but would save you a lot of time in the development process if you would fix them early. There are also ideas that actually should be in the application.

LoadSaveButton

`LoadSaveButton.cs` is the worst coded class inside the application. It's present it two scenes and in both scenes it needs a hardcoded list of prefabs. It needs the prefablist for loading in all the prefabs when loading a level. You can solve this problem easily by accessing prefabs inside the code dynamically. With the following method you can load in a prefab from the code.

-> `Resources.Load("prefabs/your prefab")`

The Inventory in the Editor

The inventory inside the Level Editor (it's actually a UIItemCollection) needs all items to be hardcoded on to it in the Unity editor. You can make this a simple dynamic process. As reference you can look at the classes `CarTaskButton.cs` and `CarTaskButtonCollection.cs`. In these classes it already has been executed successfully.

Clear Finish

Currently there is no clear sign that you finished all assigned tasks. You can solve this by writing code for the player if he doesn't have tasks left inside his TaskList.

LoadTasks & Player

Both `LoadTasks.cs` and `Player.cs` Load in the data from the datafile. Which means that if you want to change the datasource you have to change it on both places. I would advise loading data in at one of the classes and make the other class load it from that class. For example: `Player.cs` loads in data, LoadTasks loads in the data from player. With -> `FindGameObjectByTag("Player")`

Save Button does nothing

`SaveButton.cs` is empty. Currently everything gets saved automatically after the action itself. For example, you remove a block from the grid and the file overwrites the old date with the new one where that one block has been erased. A savebutton can be useful for if you decided to edit a level, but later on you decided you didn't want to edit the level. It would also be nice if you would get a warning before leaving your level without saving it first.

Eraser isn't a UIItem

The eraser itself is just a different random item. It would be nice if the eraser would also be in a UIItem, but for eraser to work you have to press on its box collider, and for the item menu to work you need to click on the box collider of the eraser. So, I don't really know how to solve this but would be nice if you can.

You can make a level without CarTasks

As the title says you can go in the editor and create a level without adding cartasks. The play section reacts weirdly to this.

You can add CarTasks that are impossible to complete

For example, adding a cartasks but not adding items to the scene. It would be nice that if you want to save a level it checks if the items required for the cartasks are present. It would be even better if you can check if the level is possible to beat before saving it.

Green Block

Does nothing you can remove it entirely. Keep in mind that it might mess with IDs of prefabs so remove it carefully.

Build as WEBGL once in a while

The app must be hosted on a website in the end. So, make sure you check once in a while if your application works the same way on the web as in unity.

Need Support?

Unity Discord Server: https://discord.gg/unity (Official Unity Discord)

Coding Discord Server: https://discord.gg/code (The Coding Den)

Ideas

Lack of Game Elements

Right now, the application is just pick up item and do a minigame. I'm sure it would be really cool if you can make the application more game like. For example, make it challenging to pick up items, maybe by making it only allowed to have 3 items at the same time so that you have to puzzle in which order you are picking up items. Make sure to discuss this with the stakeholder first.

Local Data Problem

The current idea was to open file explorer as soon as you click load level. And the same for saving a level. So that you can import and export levels into the program. This hasn't been realised due to not having enough time. If you do want to add it, discuss this with the stakeholder first. Look up the following two things -> OpenFileDialog, SaveFileDialog

Colors

Inside the program everything you can collect is lightblue. I was proposing you make all tools lightblue, and all items you can use like a wheel a different color.