# The Photographer's Guide to Los Santos

# Contents

# Preface

# Chapter 1

# Introduction

## About The Photographer's Guide to Los Santos

The Photographer's Guide to Los Santos sits between a touristic guide and a photography manual, and between an exhibition catalogue and a peak behind the scenes of artwork creation.

The Photographer's Guide to Los Santos is an ongoing project that builds on top of a research on artistic practices within spaces of computer games, with a particular focus on in-game photography, machinima and digital visual arts. It follows some themes and ideas previously explored in the exhibition *How to Win at Photography*, while focusing more specifically on the relationship between computer games and photographic activities inside the world of Grand Theft Auto V.

The idea of a guide refers to in-game photography as a form of 'virtual tourism' (Book, 2003), which was also the premise of an actual tourist guide published by Rough Guides in their 2019 *Rough Guide to XBOX*. Yet this guide project also understands the game world as a site for image production and artistic creation, turning the game into a destination for a 'game art tourist'. The Photographer's Guide to Los Santos presents the game environment of Grand Theft Auto V both as a space to explore and in which to create images, as well as a place to navigate and learn about some of the most important artworks that it has enabled to create.

The project also brings together several experiences from teaching in-game photography as an artistic practice in different educational settings and institutions, compiling materials and tools for students and artists interested in engaging with the field. The tourist guide of the game world doubles as a photography manual for the in-game photography age, featuring tutorials ranging from game screenshotting to computer programming for creative modding. Through the

practical exercises, the project invites to rethink the game object as a space for creative, subversive and critical endeavours, which can be played differently, documented, reclaimed or modified through an artistic approach.

Finally, the project draws inspiration from the works of artists who have explored the 'metaplay' of photographing game words instead of following the game rules and attempt to reach the goal of winning. The Photographer's Guide to Los Santos is indebted to all the artists it features, but was particularly inspired by Gareth Damian Martin's live streamed workshop *Photography Tour of No Man's Sky* (realized for Now Play This Festival 2020), Total Refusal & Ismaël Joffroy Chandoutis's 2021 in-game lecture performance and guided tour *Everyday Daylight* (realized for the CCS Paris), and Alan Butler's epic 2020 live endurance performance *Witness to a Changing West* (realized for Screen Walks) and his 'Content Replication Assignments'.

## Grand Theft Auto V Studies

Los Santos is the Grand Theft Auto V's fictional, parodic version of real-life Los Angeles. Just like Los Angeles is the global centre of film and commercial media production, Los Santos is the epicentre of in-game photography and machinima creation. While it may seem reductive to only focus on a single game to address the larger phenomenon of in-game photography, GTA V is the biggest source of creative outputs to date, with its extended open world and one of the largest community of active modders.

Launched in 2013, the game contains a world map of more than 80 square kilometers of total area, which includes the urban area of the city of Los Santos and the rural area of Blaine County. This incredibly vast environment features a large desert region, dense forest, several mountains, beachside towns, on top of the large metropolis of Los Santos. The game simulates the everyday life of hundreds of individual NPCs (while it allegedly counts a population of over 4 million) as well as counting 28 animal species, and more than 800 buildings in GTA V are based on real-life landmarks.

The size of the photorealistic simulation is only matched by the complexity of the game engine and its code, which - thanks to the effort of GTA V's modding community - allows players to use the game world as a powerful tool to create new scenes, take controls of its algorithmic entities, modify cameras and reshaping the game into a movie set or a photo studio.

## Grand Theft Auto V Tourism

The project can be seen as something between a playful travel guide of Los Santos and one of the star maps offered to tourist in Hollywood, pointing to the

homes of movie actors and hollywood celebrities. This guide allows players to explore the game environment following some of the most interesting artworks that have been created with(in) it. It's divided in thematic chapters that follow different artistic practices, taking place in different locations of the game environment, followed by different tutorials and exercises connected with the works and the space analyzed.

The themes explore different approaches and practices connected to established artistic and photographic currents, with a general introduction text that gives an overview of the ideas, contexts and issues connected to the specific topic. A selection of artworks for each themes is presented by a curatorial statement, introducing the work and its artistic relevance. The work is accompanied by information on the in-game location in which it was produced, inviting the readers to reach the destination in Grand Theft Auto V through maps and indications.

The game environment thus becomes the space for possible 'art tours', getting insights into the artworks made in GTA V. This form of game tourism allows the player to see the behind the scenes, and experience the making of the works in its place of origin. While the complex algorithms of GTA V produce unexpected interactions and scenes, Los Santos is also stuck in the same time forever. Gas stations, shops, palm trees remain in the same state and location forever, allowing the tourist to witness the exact scene that was first encountered by the artists.

## Grand Theft Auto V Art Education

This project is also an attempt to introduce a video game as a space for artistic intervention, and an invitation to use its mechanics, its code and its environment as a creative tool itself. The game can be played, documented and captured through a form of artistic play, that differs from normative gameplay and does not focus on advancinf and winning but rather engages with the game object critically. Furthermore, the game software can also be manipulated, modified and used as an apparatus to create new images and interactions. The goal of this guide is to combine a curatorial approach that leads the viewer to discover the artworks made in GTA V with a hands on approach that teaches the player the tools for possible artistic interventions in this space.

Games are often seen as producing specific cultures and shaping identities through through forms of play that follow the intentions of the developer. Here we understand games as objects to be reclaimed and tools to be deconstructed and rebuilt, both conceptually and literally. Consequently, players are not just passive actors that push buttons in the sequence that they are taught by the machine and its softwares, but open up the black box of the game and become critical thinkers and makers that actively play with the game, or even against it.

The Photographer's Guide to Los Santos can be employed as a resource to accompany workshops for students and artists approaching computer games and interested in learning how to engage with it. Each thematic chapter features a tutorial section that introduces different techniques and strategies to capture images within Grand Theft Auto V, connected to the examples and locations of each section. The chapters are thought to be experienced in order, as the tutorials at times rely on knowledge that is built on top of previous lessons. Each tutorial is accompanied by content replication assignments, in which the readers is invited to use the skills learnd from each chapter to recreate a work presented in that section. Tutorials are intended for anyone who might be playing GTA V for the first time in their life and do not assume any previous experience, although some basic idea of programming is helpful when dealing with scripting and modding the game.

# Chapter 2

# Architectural Photography

//general intro on architecture in game spaces, the contruction of virtual cities, architecture photography and how it relates to the game environment, the player as a photographer documenting urban spaces…

> "I saw it as the end of architecture… by pushing the concept to its limit and primarily by using the photograph as a point of departure. This is reflected in the idea that the great majority of images are no longer the expression of a subject, or the reality of an object, but almost exclusively the technical fulfilment of all its intrinsic possibilities. It's the photographic medium that does all the work. People think they're photographing a scene, but they're only technical operators of the device's infinite virtuality. The virtual is the device that wants nothing more than to function, that demands to function." – Jean Baudrillard - *The Singular Objects of Architecture and Philosophy*, 2000

Architecture photography was born with the invention of heliographs, daguerrotypes and large format cameras in the first half of the 19th century. Due to the long exposure times, buildings were the ideal subjects for the early scientific experiments of oseph Nicéphore Niépce, Louis-Jacques-Mandé Daguerre and William Talbot.

*View from the Window at Le Gras* by Joseph Nicéphore Niépce, 1826-27

*Paris' Boulevard du Temple* by Louis-Jacques-Mandé Daguerre, 1839

While the relationship between architecture and photography has been part of the medium from its birth for technical reasons, this form of image making has evolved to visually explore the connection with material spaces and forms, as well as the relation between human perception and architectural bodies.

The photographic image is not simply a document of a structure, but "is, in fact, *part* of its architecture".[1] Curator Urs Stahel wrote that "pictures [...] offer a discourse that is unlike the physical experience of architecture. They transform volume into surface; distil matter into forms and signs. Photography shapes architecture, enlarging and reducing it, heightening and shortening it, accentuating it, yet largely leaving it to its own devices."[2]

The first architectural photographer is considered to be Joseph-Philibert Girault de Prangey (1804 – 1892), who started to take daguerrotypes of iconic buildings like the Parthenon in Athens and Notre Dame in Paris from 1841. Architectural photography evolved in two distinct approaches, namely Elevation and Perspective. The Elevation Approach focuses on representing a structure as a two-dimensional image, obtaining a viewpoint that is parallel to the building and aimed at showing as many details as possible. The Perspective Approach aims at depicting the structure within the space, focusing on the third dimension and often taken at an angle or from a vantage point from a corner.

*Cathédrale Notre-Dame de Paris* by Joseph-Philibert Girault Prangey, 1841

//Architecture photography and the modernist project //Modern architecture started to flourish with photographs in about the 1920s when urban photographers like Eugène Atget, Berenice Abbott, Walker Evans, and T. Lux Feininger-came into play.

//Eugène Atget(12 February 1857 – 4 August 1927), a French photographer known for his documentation of the streets of Paris before it entered Modernization. His works were published by Berenice Abbott after his death.

//Berenice Abbott(July 17, 1898 – December 9, 1991), an American Photographer is known for New York and urban design photographs in the 1930s and also for her portraits of between-the-wars 20th century cultural figures.

//"Modernist architecture and photography have been ideologically interconnected" Lorenzo Rocha //close connection between photographer and architect: //Armando Salas Portugal for Luis Barragán, //Bill Engdahl for Mies van der Rohe, //Julius Shulman for Richard Neutra

//Bernhard Becher and Hilla Becher, a German conceptual artist and photographer also known as Bernd and Hilla Becher. They worked on Duo in projects which include photography of industrial buildings and structures and they were often organized in grids. They wrote several books and their works are for public display in the Art Institute of Chicago, Chicago, Tate Gallery, London, Museum of Modern Art, New York, and several other famous museums.

//Michael Wesely, a German Architectural photographer known for his ultra-long exposure shots //Potsdamer Platz, Berlin (1997-99) by Michael Wesely

[1] Lorenzo Rocha, *Photography and Modern Architecture* ,"Concrete - Photography and Architecture", Scheidegger & Spiess, 2013

[2] Urs Stahel, Foreword to "Concrete - Photography and Architecture", Scheidegger & Spiess, 2013

//Hélène Binet postmodern architecture

//CGI //Into the universe of rendered architectural images - Joel McKim //Rendering the Desert of The Real – Tobias Revell //The Entasis of Elon Musk – Tamar Shafrir

//Game Urbanism https://www.youandpea.com/atlas heterotopias

"We are outnumbered by virtual worlds, overwhelmed by virtual architecture. Videogames and digital art have furnished us with a hundred thousand matterless forms–landscapes where no rock or earth has ever been present, cities founded on depthless skins of image and texture, expanses that will never see the light of a true sun. And yet, somehow there is material here, a new kind of matter. Some of it is borrowed—photographs, texture references, photogrammetry. Other parts are inherent properties of digital worlds— their obsession with surface, the logic of their light, their base particles; pixels, voxels, polygons."[3]

## *The Continuous City*, by Gareth Damian Martin

Gareth Damian Martin, *Outskirts*, from *The Continuous City*,

Gareth Damian Martin, *Pathways*, from *The Continuous City*,

artwork text

More about *The Continuous City*

Interview with Gareth Damian Martin

### Getting there

- The intersection of Interstate 4 and Interstate 5 manifests the architecture of traffic of the megalopolis.

## Readings

Heterotopias

Mark D Teo, The Urban Architecture of Los Angeles and Grand Theft Auto, 2015. https://www.academia.edu/18173221/The_Urban_Architecture_of_Los_Angeles_and_Grand_Theft_Auto

[3]Gareth Damian Martin, *Represented , Contested, Inverted*, in "Heterotopias 001", 2017

# Tutorial

## Photographing the Game Screen

### Analogue Game Photography

//Photographs of a TV screen taken with a digital camera often exhibit moiré patterns. Since both the TV screen and the digital camera use a scanning technique to produce or to capture pictures with horizontal scan lines, the conflicting sets of lines cause the moiré patterns. To avoid the effect, the digital camera can be aimed at an angle of 30 degrees to the TV screen.

### Screenshotting

On windows there are several ways to take a screenshot. To capture your entire screen and automatically save the screenshot, press the `Windows logo key` + `PrtScn key` (or `fn key` + `Windows logo key` + `PrtScn key`). The screenshot will be saved to `Pictures > Screenshots folder`.

On windows 10 and 11 you can use the Game bar to take game screenshots and start/stop game screen recordings. Press the `Windows logo key` + `G` on your keyboard to open Game Bar.

- Press the camera icon to take a screenshot of the game screen.
- Press the circle icon to start a clip, then the square icon to stop recording the game screen.
- Click on “See my captures” to access the image and video files.

# Content Replication Assignment

# Chapter 3

# Social Documentary

//general intro on simulating society, the creation of NPCs, documentary and street photography traditions connected to politics of visibility and representation, and how they relate to the politics of simulation, how the player-photographer documents the creation of complex social spaces and reveals the process of simulating people and issues of class, gender, race in the game space…

## *Down and Out in Los Santos* by Alan Butler

artwork text

More about *Down and Out in Los Santos*

### Getting There

The homeless camp in Los Santos is under the Olympic Freeway in Strawberry.

Dignity Village is a tent city established by homeless people near Procopio Beach, east of Paleto Bay.

## *Fear and Loathing in GTA V* by Morten Rockford Ravn

artwork text

More about *Fear and Loathing in GTA V*

### Getting There

## Readings

## Tutorial

### In-game Smartphone Camera

Snapmatic is the photo app on your simulated mobile phone in GTA V.

- Press `UP` on the on the keyboard (PC) or d-pad (Playstation) to bring up your phone.
- Select the Snapmatic app - it's on the bottom left of the homescreen.
- Move the camera with the `Mouse` on PC, or with the `RIGHT STICK` on Playstation.
- Zoom in and out with the `Mouse Wheel` on PC, or `LEFT STICK` on Playstation.
- You can shuffle through filters with `DOWN` or borders with `UP`.
- To take selfie press the `Mouse Wheel Button` on PC or `R3 STICK` on Playstation to turn the camera on yourself.
- Once you're happy, take the photo with "È`nteron pc or`X"' on the Playstation. Press it again to save it to the Gallery.
- You can upload your picture to your Rockstar Game sSocial Club profile by going to the gallery and pressing the "Left Ctrl" key on PC.
- Your photos will be published on socialclub.rockstargames.com/member/USERNAME/photos, where USERNAME is replaced by your actual username.

## Content Replication Assignment

# Chapter 4

# Re-enactment Photography

//general introduction on the development of photorealism in games, the relationship between photography and CGI, the remediation of photographic images and the analog apparatus, the player as photographer situated in the tradition of conceptual photographers like Sherrie Levine and Sturtevant, the copy as a conceptual approach that create new meaning through a similar image but a different context…

## *26 Gasoline stations in GTA V* by Lorna Ruth Galloway

artwork text

More about *26 Gasoline stations in GTA V*

### Getting There

- Globe Oil Gas Station, Innocence Blvd & Alta St, South Los Santos
- LTD Gas Station, Davis Ave & Grove St, South Los Santos
- LTD Gas Station, Mirror Park Blvd & W Mirror Dr, Mirror Park
- Globe Oil Gas Station, Clinton Ave & Fenwell Pl, Vinewood Hills
- Xero Gas Station, Strawberry Ave & Capital Blvd, South Los Santos
- RON Gas Station, Davis Ave & Macdonald St, South Los Santos
- Xero Gas Station, Calais Ave & Innocence Blvd, Little Seoul
- LTD Gas Station, Lindsay Circus & Ginger St, Little Seoul
- RON Gas Station, N Rockford Dr & Perth St, Morningwood
- Xero Gas Station, Great Ocean Hwy, Pacific Bluffs

### *A Study on Perspective* by Roc Herms

artwork text

More about *A Study on Perspective*

#### Getting There

Vinewood Sign, Vinewood Hills

## Further references

### *Little Books of Los Santos* by Luke Caspar Pearson

artwork text

More about *Little Books of Los Santos*

### *26 Gasoline stations in GTA V* by M. Earl Williams

artwork text

More about *26 Gasoline stations in GTA V*

## Readings

## Tutorial

#### Scene Director Mode

## Content Replication Assignment

# Chapter 5

# Nature Documentary

//general introduction about the creation of a synthetic forms of nature, ecological issues, creation of virtual sublime, flora and fauna that are usually props that become the focus of the player's explorations, "virtual world naturalism"...

## *San Andreas Streaming Deer Cam* by Brent Watanabe

artwork text

More about *Deercam*

### Getting There

Mount Chiliad is located in the Chiliad Mountain State Wilderness, and it is the tallest mountain in the game at 798m above sea level. The state park is home to lots of wildlife such as deer and mountain lions.

## *Virtual Botany Cyanotype* by Alan Butler

//selection of flora from GTA V

artwork text

More about *Virtual Botany Cyanotype*

### Getting There

## Readings

## Tutorial

### Scripting Introduction

### Preparation and Setup

- Install Windows 11
- Download and install Steam (with a copy of GTA V or buy the game if you do not have it. GTA V is 100+ GB so it will take a few hours depending on your internet connections)
- Download Script Hook V, go to the bin folder and copy `dinput8.dll` and `ScriptHookV.dll` files into your GTA V directory `C:\Program Files (x86)\Steam\steamapps\common\Grand Theft Auto V`
- Download Script Hook V dot net, copy the `ScriptHookVDotNet.asi` file, `ScriptHookVDotNet2.dll` and `ScriptHookVDotNet3.dll` files into your GTA V directory `C:\Program Files (x86)\Steam\steamapps\common\Grand Theft Auto V`
- Create a new folder in GTA V directory and call it "scripts".
- Download and install Visual Studio Community (free version of VS). Open Visual Studio and check the .NET desktop development package and install it
- Run GTA V and test if Script Hook V is working by pressing `F4`. This should toggle the console view. Try to type Help() and press "ÈNTER"' to get a list of available commands.
- It's also recommended to use a completed game save file to skip the story mode and compulsory introductory mission. Extract the content of the file and copy both files `SGTAXXXXX` and `SGTAXXXXX.bak` in `Documents/GTA V/Profiles/YYYYYYYY/`. Once the files are there, you will be able to load the game state by running GTA V and navigating to GAME > Load Game and select the 100% game file.

#### Creating a Mod File

- Open Visual Studio

- Select File > New > Project
- Select Visual C# and Class Library (.NET Framework)
- Give a custom file name (e.g. moddingTutorial)
- Rename public class Class1 as “moddingTutorial” in the right panel Solution Explorer and click Yes on the pop-up window
- In the same panel go to References and click add References…
- Click on > Browse > browse to Downloads
- Select ScriptHookedVDotNet > `ScriptHookVDotNet2.dll` and `ScriptHookVDotNet3.dll` and add them
- Go to > Assemblies and search for “forms” and select `System.Windows.forms`
- Search for “drawing” and select `System.Drawing`
- In your code file add the following lines on top:

```
using GTA;
using GTA.Math;
using System.Windows.Forms;
using System.Drawing;
using GTA.Native;
```

- Modify class moddingTutorial to the following:

```
namespace moddingTutorial
{
  public class moddingTutorial : Script
  {
      public moddingTutorial()
      {
          this.Tick += onTick;
      this.KeyUp += onKeyUp;
      this.KeyDown += onKeyDown;
      }

      private void onTick(object sender, EventArgs e)
      {
      }

      private void onKeyUp(object sender, KeyEventArgs e)
      {
      }
```

```
      private void onKeyDown(object sender, KeyEventArgs e)
      {
        if (e.KeyCode == Keys.H)
        {
              Game.Player.ChangeModel(PedHash.Cat);
          }
      }
  }
}
```

- Save file
- Go to Documents > Visual Studio > Project > moddingTutorial > moddingTutorial > `moddingTutorial.cs`
- Copy the .cs file in the GTA V directory inside the scripts folder
- Open GTA V, run the game in Story Mode (mods are only allowed in single player mode, not in GTA Online) and press 'H' to see if the game turns your avatar into a cat
- Note: every time you make changes to your .cs file in the scripts folder you can hit `F4` to open the console, type `Reload()` in the console for the program to reload the script and test again the changes.

### onTick, onKeyUp and onKeyDown

The main events of Script Hook V Dot Net are onTick, onKeyUp and onKeyDown. Script Hook V Dot Net will invoke your functions whenever an event is called.

The code within the onTick brackets is executed every interval milliseconds (which is by default 0), meaning that the event will be executed at every frame, for as long as the game is running.

```
private void onTick(object sender, EventArgs e)
{
       //code here will be executed every frame (or per usef defined interval)
}
```

If your function is written inside onKeyDown (withiin the curly brackets following onKeyUp(object sender, KeyEventArgs e){}), your code will be executed every time a key is pressed. If your function is written inside onKeyUp, your code will be executed every time a key is released.

```
private void onKeyUp(object sender, KeyEventArgs e)
{
      //code here will be executed whenever a key is released
}

private void onKeyDown(object sender, KeyEventArgs e)
{
      //code here will be executed whenever a key is pressed
}
```

We can specify which code is executed based on what keys are pressed/released

```
private void onKeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.H)
    {
        //code here will be executed whenever the key 'H' is pressed
    }
}
```

## Change Player Model

The player character is controlled as `Game.Player`. `Game.Player` can perform different functions, including changing the avatar model.

Change the 3D model of your character by using the `ChangeModel` function. The function needs a model ID, in order to load the model file of our game character. You can browse through this list of models to find the one you want to try (note: not all models seem to load properly): wiki.gtanet.work/index.php/Peds

These models are all PedHashes, basically ID numbers within the PedHash group. Copy the name of the model below the image and add it to PedHash. For example if you choose the model Poodle, you'll need to write `PedHash.Poodle`.

To change the model of your player character into a poodle you can write the following function:

```
Game.Player.ChangeModel(PedHash.Poodle);
```

add it in your .cs file in the onKeyDown event, triggered by the pressing of the 'h' key:

Example code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using GTA;
using GTA.Math;
using System.Windows.Forms;
using System.Drawing;
using GTA.Native;


namespace moddingTutorial
{
    public class moddingTutorial : Script
    {
        public moddingTutorial()
        {
            this.Tick += onTick;
            this.KeyUp += onKeyUp;
            this.KeyDown += onKeyDown;
        }

        private void onTick(object sender, EventArgs e) //this function gets executed
        {
        }

        private void onKeyUp(object sender, KeyEventArgs e)//everything inside here is
        {
        }

        private void onKeyDown(object sender, KeyEventArgs e) //everything inside here
        {
            //when pressing 'H'
            if(e.KeyCode == Keys.H)
            {
                //change player char into a different model
                Game.Player.ChangeModel(PedHash.Poodle);
            }
        }
    }
}
```

Try to select different models and assign them to different keys to change the

model of your character. Use keys that are not already implemented in the game controls to avoid clashes with built in operations.

## Tasks

Our character can be controlled by our script, and given actions that override manual control of the player. These actions are called *Tasks* and in order to assign tasks to our characters we have to define our `Game.Player` as `Game.Player.Character`. The `Game.Player.Character` code gets the specific model the player is controlling.

Now we can give tasks to the character by adding the `Task` function: `Game.Player.Character.Task`.

Finally we can specify what task to give the character by choosing a task from TaskInvoker list of possible actions.

Jump:

```
Game.Player.Character.Task.Jump();
```

Wander around:

```
Game.Player.Character.Task.WanderAround();
```

Hands up for 3000 milliseconds:

```
Game.Player.Character.Task.HandsUp(3000);
```

Turn towards the camera:

```
Game.Player.Character.Task.TurnTo(GameplayCamera.Position);
```

Some of the tasks are temporary and accept a time parameter (in milliseconds). Others are persistent, meaning they will keep being executed until the task is actively stopped. To stop a task you can use the `ClearAllImmediately();` command:

```
Game.Player.Task.ClearAllImmediately();
```

### Task Sequences

You can create sequence of multiple tasks by using `TaskSequence` and the `PerformSequence` function. Create a new `TaskSequence` with a custom name, add tasks to it with `AddTask`, close the sequence with `Close` and then call `Task.PerformSequence` to perform the sequence.

```
TaskSequence mySeq = new TaskSequence();
mySeq.AddTask.Jump();
mySeq.AddTask.HandsUp(3000);
mySeq.Close();

Game.Player.Character.Task.PerformSequence(mySeq);
```

### Random

We can add randomness by using a randomly generated number, which makes things outside of thepredefined programme controlled by us and introduces more autonomous behaviours. We use the `Random`function to create a randomly generated number between our minimum and maximum parameter (if only one parameter is inserted, the minimum is 0).

```
Random rnd = new Random();
int month = rnd.Next(1, 13); // creates a number between 1 and 12
int dice = rnd.Next(1, 7); // creates a number between 1 and 6
int card = rnd.Next(52); // creates a number between 0 and 51
```

Let's create a number to generate a random duration between 1 and 6 seconds, for the `HandsUp` task.

```
Random rnd = new Random();
int waitingTime = rnd.Next(1, 7);
Game.Player.Character.Task.HandsUp(waitingTime * 1000);
```

### Subtitles and Notifications

Generate subtitles with a custom text string and duration (in milliseconds):

```
UI.ShowSubtitle("Hello World", 3000);
```

Generate a notification with a custom text string:

```
UI.Notify("Hello World");
```

# Content Replication Assignment

## Deercam reenactment

Write a mod script to change your game character into a deer by pressing a key, and make it autonomously wander around Los Santos by pressing another key.

# Chapter 6

# Surrealist Photography

//general introduction about avant garde traditions of distancing from reality and exploring the possibilities of CGI decoupled from realism adn life-like simulation, the game as an engine that can be used to create oniric scenes, which in turn reveal the untapped possibilities hidden within the game code, the player as a modder which can generate worlds within the world…

## Alexey Andrienko aka HAPP v2

artwork text

More about Happ v2

### Getting There

Chumash Beach

## Readings

## Tutorial

### Scripting Characters

### NPCs

NPCs are non playable characters and in GTA V scripting they are called `Peds`. Peds are an entity like Props or Vehicles and can be created, assigned different

model textures, equipped with weapons and controlled through different tasks.

## Spawn a new NPC

A GTA V Ped can be created by the `World.CreatePed` function. This takes two parameters: an ID to assign the 3D model and textures, and the location where the Ped is created.

The model IDs are the same we used in the previous tutorial, when we changed our character's appearance to a cat. A list of all available models can be found here. `PedHash.Cat`, `PedHash.Deer`, `PedHash.AviSchwartzman`are all possible IDs we can assign to the NPC we want to create. We can create a new model variable, which we will name 'myPedModel' and assign it a model ID:

```
Model myPedModel = PedHash.AviSchwartzman;
```

The location where the NPC is created through a vector3 data type, which represents a vector in 3D space. This basically means a point that contains X, Y and Z coordinates. We can give absolute coordinates, making the Ped appear at a specific location in the game, but we can also use a location relative to our position in the game. In order not to risk making a Ped appear somewhere completely outside of our view – on some mountain or in the sea – let's look at a vector3 that points to a position in front of the player.

We want to establish the player with`Game.Player.Character`, followed by a function that retireve the player position within the game world. That's called by using `GetOffsetInWorldCoords`, which takes a vector3. The values of the X, Y and Z of the vector 3 offset the location based on the origin point represented by the player. Therefore, we can move the place where we want the Ped to appear by adding values to the X axis (left or right of player), Y axis (ahead or behind the player), and Z axis (above or below the player). To make a Ped appear in front of the player we can create a vector3 data type with 0 for X, 5 for Y and 0 for Z: `new Vector3(0, 5, 0)`. Let's make a vector3 variable, which we will name 'myPedSpawnPosition', assign it the values above for X, Y and Z coordinates from the player position.

```
Vector3 myPedSpawnPosition = Game.Player.Character.GetOffsetInWorldCoords(new Vector3(0, 5, 0));
```

Now we can use the model and the position variables to spawn the NPC in front of the player. We'll create a Ped named 'myPed1' and use the `World.CreatePed` function with the two variables as parameters:

```
var myPed1 = World.CreatePed(myPedModel, myPedSpawnPosition);
```

Example code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using GTA;
using GTA.Math;
using System.Windows.Forms;
using System.Drawing;
using GTA.Native;


namespace moddingTutorial
{
    public class moddingTutorial : Script
    {
        public moddingTutorial()
        {
            this.Tick += onTick;
            this.KeyUp += onKeyUp;
            this.KeyDown += onKeyDown;
        }

        private void onTick(object sender, EventArgs e) //this function gets executed
        {
        }

        private void onKeyUp(object sender, KeyEventArgs e)//everything inside here is
        {
        }

        private void onKeyDown(object sender, KeyEventArgs e) //everything inside here
        {
            //when pressing 'K'
            if(e.KeyCode == Keys.K)
            {
                //select a model and store it in a variable
                Model myPedModel = PedHash.AviSchwartzman;

        //create a position relative to the player
        Vector3 myPedSpawnPosition = Game.Player.Character.GetOffsetInWorldCoords(new

        //create a Ped with the chosen model, spawning at the chosen position
        var myPed1 = World.CreatePed(myPedModel, myPedSpawnPosition);
```

```
            }
        }
    }
}
```

## Control Multiple NPCs

You can create multiple NPCs and give them custom names. Let's create a human NPC and a cat NPC and call them Jim and MannyTheCat respectively:

```
var Jim = World.CreatePed(PedHash.AviSchwartzman, Game.Player.Character.GetOffsetInWorldCoords(ne
var MannyTheCat = World.CreatePed(PedHash.Cat, Game.Player.Character.GetOffsetInWorldCoords(new V
```

Try to kill one of the `Ped` NPCs you created by using the `Kill()`.

```
Jim.Kill();
```

Note that when you kill your `Ped` 'Jim', it falls on the floor and it won't actually respond to any call or task you will give it, but it's not removed from the game. To remove a specific `Ped` you have to use the `Delete`function, which will remove that instance (and will make the NPC disappear).

```
Jim.Delete();
```

To handle groups of NPCs we can use the `List` class. A `List` is a collection of objects, and a `List`of `Peds` allows us to store our NPCs. We can use an index to retrieve and control specific `Peds` in the group. You can see the reference for more detailed information.

Create a `List` of `Peds` named myPeds as a global variable in the public class `public class moddingTutorial : Script`.

```
List<Ped> myPeds = new List<Ped>();
```

In the onKeyDown function `private void onKeyDown(object sender, KeyEventArgs e)` create 5 new Peds with a For Loop

```
for (int i = 0; i < 5; i++)
{
    //spawn a new Ped called newPed
    var newPed = World.CreatePed(PedHash.Clown01SMY, Game.Player.Character.GetOffsetInWorldCoords
    //add the new Ped to my list of Peds myPeds
    myPeds.Add(newPed);
}
```

Now all the 5 `Peds` are part of the myPeds[] `List`. You can control each Ped individually by calling their individual number ID in the group. The first spawn `Ped` is myPed[0], the last one is myPeds[4].

Tell the 1st spawned NPC to start wandering around:

```
myPeds[0].Task.WanderAround();
```

Kill the 2nd spawned NPC:

```
myPeds[1].Kill();
```

Tell the 3rd NPC to jump:

```
myPeds[2].Jump();
```

Tell the 4th NPC to walk toward the camera:

```
myPeds[3].GoTo(GameplayCamera.Position);
```

Tell the 5th NPC to put their hands up for 3 seconds:

```
myPeds[4].Task.HandsUp(3000);
```

## Nearby NPCs

Script Hook V DOt Net provides a function `GetNearbyPeds`which groups all the `Peds` within a nearby radius from a character.

Create a new group that adds `Peds` which are closer than 20 meters from the player (add that as a global variable in `public class moddingTutorial : Script`):

```
Ped[] NearbyPeds = World.GetNearbyPeds(Game.Player.Character, 20f);
```

Use a Foreach Loop to get every `Ped` in the group and give them the task to put their hands up for a second:

```
foreach (Ped p in NearbyPeds)
{
    p.Task.HandsUp(1000);
}
```

`GetNearbyPeds` does not sort out individual `Pedsin` the group based on distance, so we have to do a bit of manual filtering to get the nearest NPC within the chosen radius from the player character.

Define the global variables in the public class `public class moddingTutorial : Script`:

```
float lastDistance;
Ped nearestPed = null;
Ped oldNearestPed = null;
```

Get and parse the nearby NPCs in the OnTick function `private void onTick(object sender, EventArgs e)`:

```
//set radius
float maxDistance = 25f;
//get nearest peds
Ped[] pedsGroup = World.GetNearbyPeds(Game.Player.Character, maxDistance);

float lastDistance = maxDistance;
foreach (Ped ped in pedsGroup)
{
    float distance = ped.Position.DistanceTo(Game.Player.Character.Position);
    if (distance < lastDistance)
    {
        nearestPed = ped;
        lastDistance = distance;
    }
}

    if (nearestPed != null && oldNearestPed != nearestPed)
    {
        nearestPed.Task.HandsUp(1000);
    }
oldNearestPed = nearestPed;
```

## Give Tasks to NPCs

A Ped can be given a task using the `Task` function, just like we did in the previous tutorial for the player character.

```
myPed1.Task.WanderAround();
```

Some tasks involve interacting with other characters (Peds or Game.Player.Character) or take different parameters like positions (vector3), duration (in milliseconds),

and other data types. We can give our NPC the task to fight against the player by using the `FightAgainst` function, which requires a Ped parameter – which in the case of the player is expressed as `Game.Player.Character`.

```
myPed1.Task.FightAgainst(Game.Player.Character); //give npc task to fight against play
```

Try to replace the task to "fight against" with "flee from (player)" , "hands up", "jump"… or some of the other available tasks.

See the TaskInvoker list for possible tasks, or click on the list of available tasks below.

List of Available Tasks

```
void AchieveHeading (float heading, int timeout=0)

void AimAt (Entity target, int duration)

void AimAt (Vector3 target, int duration)

void Arrest (Ped ped)

void ChatTo (Ped ped)

void Jump ()

void Climb ()

void ClimbLadder ()

void Cower (int duration)

void ChaseWithGroundVehicle (Ped target)

void ChaseWithHelicopter (Ped target, Vector3 offset)

void ChaseWithPlane (Ped target, Vector3 offset)

void CruiseWithVehicle (Vehicle vehicle, float speed, DrivingStyle style=DrivingStyle.

void DriveTo (Vehicle vehicle, Vector3 target, float radius, float speed, DrivingStyle

void EnterAnyVehicle (VehicleSeat seat=VehicleSeat.Any, int timeout=-1, float speed=1f

void EnterVehicle (Vehicle vehicle, VehicleSeat seat=VehicleSeat.Any, int timeout=-1,
```

```
void FightAgainst (Ped target)

void FightAgainst (Ped target, int duration)

void FightAgainstHatedTargets (float radius)

void FightAgainstHatedTargets (float radius, int duration)

void FleeFrom (Ped ped, int duration=-1)

void FleeFrom (Vector3 position, int duration=-1)

void FollowPointRoute (params Vector3[] points)

void FollowPointRoute (float movementSpeed, params Vector3[] points)

void FollowToOffsetFromEntity (Entity target, Vector3 offset, float movementSpeed, int timeout=-1

void GoTo (Entity target, Vector3 offset=default(Vector3), int timeout=-1)

void GoTo (Vector3 position, int timeout=-1)

void GoStraightTo (Vector3 position, int timeout=-1, float targetHeading=0f, float distanceToSlid

void GuardCurrentPosition ()

void HandsUp (int duration)

void LandPlane (Vector3 startPosition, Vector3 touchdownPosition, Vehicle plane=null)

void LeaveVehicle (LeaveVehicleFlags flags=LeaveVehicleFlags.None)

void LeaveVehicle (Vehicle vehicle, bool closeDoor)

void LeaveVehicle (Vehicle vehicle, LeaveVehicleFlags flags)

void LookAt (Entity target, int duration=-1)

void LookAt (Vector3 position, int duration=-1)

void ParachuteTo (Vector3 position)

void ParkVehicle (Vehicle vehicle, Vector3 position, float heading, float radius=20.0f, bool keep

void PerformSequence (TaskSequence sequence)
```

```
void PlayAnimation (string animDict, string animName)

void PlayAnimation (string animDict, string animName, float speed, int duration, float

void PlayAnimation (string animDict, string animName, float blendInSpeed, int duration

void PlayAnimation (string animDict, string animName, float blendInSpeed, float blendOu

void RappelFromHelicopter ()

void ReactAndFlee (Ped ped)

void ReloadWeapon ()

void RunTo (Vector3 position, bool ignorePaths=false, int timeout=-1)

void ShootAt (Ped target, int duration=-1, FiringPattern pattern=FiringPattern.Default)

void ShootAt (Vector3 position, int duration=-1, FiringPattern pattern=FiringPattern.De

void ShuffleToNextVehicleSeat (Vehicle vehicle=null)

void Skydive ()

void SlideTo (Vector3 position, float heading)

void StandStill (int duration)

void StartScenario (string name, float heading)

void StartScenario (string name, Vector3 position, float heading)

void SwapWeapon ()

void TurnTo (Entity target, int duration=-1)

void TurnTo (Vector3 position, int duration=-1)

void UseParachute ()

void UseMobilePhone ()

void UseMobilePhone (int duration)

void PutAwayParachute ()
```

```
void PutAwayMobilePhone ()

void VehicleChase (Ped target)

void VehicleShootAtPed (Ped target)

void Wait (int duration)

void WanderAround ()

void WanderAround (Vector3 position, float radius)

void WarpIntoVehicle (Vehicle vehicle, VehicleSeat seat)

void WarpOutOfVehicle (Vehicle vehicle)

void ClearAll ()

void ClearAllImmediately ()

void ClearLookAt ()

void ClearSecondary ()

void ClearAnimation (string animSet, string animName)
```

You can spawn a group of NPCs and give them individual tasks. You can also make them interact with each other (or with the player character). Here we spawn 3 NPCs and tell the to fight with each other.

```
//create a list of Peds
List<Ped> myPeds = new List<Ped>();

//create a list of Ped models
List<Model> myPedModel = new List<Model>();

//manually add models for each ped
myPedModel.Add(PedHash.Clown01SMY);
myPedModel.Add(PedHash.Doctor01SMM);
myPedModel.Add(PedHash.Abigail);


for(int i = 0; i < myPedModel.Count; i++)
{
```

```
    //spawn a new Ped for each model
    var newPed = World.CreatePed(myPedModel[i], Game.Player.Character.GetOffsetInWorldC
    //add the new Ped to my list of Peds
    myPeds.Add(newPed);
}

myPeds[0].Task.FightAgainst(myPeds[1]);
myPeds[1].Task.FightAgainst(myPeds[2]);
myPeds[2].Task.FightAgainst(myPeds[0]);
```

To clear a task at any given moment we can use the task `ClearAllImmediately();`. To stop our 3 NPCs from fighting each other we give them the task to stop everything they are doing immediately.

```
myPeds[0].Task.ClearAllImmediately();
myPeds[1].Task.ClearAllImmediately();
myPeds[2].Task.ClearAllImmediately();
```

Peace is restored in the universe. To remove the NPCs use `Delete()`.

```
myPeds[0].Delete();
myPeds[1].Delete();
myPeds[2].Delete();
```

## Animations

We can give Peds a task to play specific animations. To do this we can use the native function `TASK_PLAY_ANIM`. The function takes a lot of parameters (some of them still not exactly know), but here is the full function and a breakdown of each parameters.

```
Native.Function.Call(Native.Hash.TASK_PLAY_ANIM, thePed, sDict, sAnim, speed, speed * -
```

`thePed` The Ped that will play the animation `sDict` The dictionary where the anim is located `sAnim` The anim name `speed` The play start speed (This is important to make smooth changes between anims) `speed * -1` Unknown `-1` Unknown `flags` Flags that you can set for the playback (see flags below) `0` Unknown `false` Unknown `bDisableLegIK` If the anim will ignore the leg/foot interaction with obstacles `false` Unknown

Flags for playback modes

```
```

```
normal = 0
    repeat = 1
    stop_last_frame = 2
    unk1 = 4
    unk2_air = 8
    upperbody = 16
    enablePlCtrl = 32
    unk3 = 64
    cancelable = 128
    unk4_creature = 256
    unk5_freezePos = 512
    unk6_rot90 = 1024
```

You need to request the animation dictionary before start using it in your script: `REQUEST_ANIM_DICT`. After that, wait for the animation to load (or you could check if it's loaded with the boolean `HAS_ANIM_DICT_LOADED`), before playing the animation.

Once you have requested your animation dictionary and it is loaded, you can play and stop the specific animation using `TASK_PLAY_ANIM` and `STOP_ANIM_TASK`.

```
//request animation dictionary
Function.Call(Hash.REQUEST_ANIM_DICT, "cmini@strip_club@pole_dance@pole_a_2_stage");

//wait 100 ms to load the animation
Wait(100);

//play animation from animation dictionary using the player character
Function.Call(Hash.TASK_PLAY_ANIM, Game.Player.Character, "mini@strip_club@pole_dance@pole_a_2_st

//wait 5 secs
Wait(5000);

//stop the animation
Function.Call(Hash.STOP_ANIM_TASK, Game.Player.Character, "mini@strip_club@pole_dance@pole_a_2_st
```

Most information found for this functions were found here. More example code and information is avaiable there.

There are 6645 animation dictionaries and 35460 animation clips. You can see some of the possible animations in GTA V here: youtube.com/playlist?list=PLFy_1HUkWwEAgPtwtjjLYpKCbBiwX
Here you can find a list of available dictionaries and animations.

Fun fact: deers seem to be able to do pole dance animations too.

Give animations to nearby peds.

```
//request animation dictionary
Function.Call(Hash.REQUEST_ANIM_DICT, "gestures@miss@fbi_5");

//wait for it to load
Wait(50);

//get nearby ped
Ped[] NearbyPeds = World.GetNearbyPeds(Game.Player.Character, 20f);

foreach (Ped p in NearbyPeds)
{
    //clear the peds of any tasks they might have
    p.Task.ClearAllImmediately();
    //play animation from animation dictionary
    Function.Call(Hash.TASK_PLAY_ANIM, p, "missfbi5ig_2", "crying_trevor", 8.0, 8.0 * -
}
```

## Teleporting

We can change the location of the player character or of any Ped or Vehicle entity by using the native function `SET_ENTITY_COORDS`. This function needs an entity and X, Y and Z coordinate to teleport to. We need to know the exact coordinates of the locations we want to teleport to, but thankfully the modding community forums provide lists with all available coordinates we can teleport to. Let's take the XYZ coordinates of the top of Mount Chiliad (the highest point in the game) to teleport our player character to.

```
LOCATION: Top of the Mt Chilad
COORDINATES: X:450.718 Y:5566.614 Z:806.183
```

To create a teleport function we will use a native function. Script Hook V Dot Net is a wrapper for the C++ ScriptHook, calling the functions in Scripthook to do things in the game. However, there are some functions that are not in Script Hook V Dot Net and in order to use these, we have to use the native calling from Script Hook.

Native functions are called with `Function.Call` followed by their corresponding hash name and parameters. They use this structure:

```
Function.Call(Hash.HASH_NAME, input_params);
```

The native function for teleporting expects the hash `SET_ENTITY_COORDS`, the `ped` entity to teleport, and the X, Y and Z coordinates to teleport the character to. `Function.Call(Hash.SET_ENTITY_COORDS, Ped ped, X, Y, Z, 0, 0, 1);`

The function to teleport the player character to the top of Moutn Chiliad is:

```
//Teleport to the top of Mount Chiliad
Function.Call(Hash.SET_ENTITY_COORDS, Game.Player.Character, 450.718f, 5566.614f, 806.183f, 0, 0,
```

See this list of locations to find their respective coordinates or click on the list below

List of Locations with Coordinates

INDOOR LOCATIONS

Strip Club DJ Booth X:126.135 Y:-1278.583 Z:29.270

Blaine County Savings Bank X:-109.299 Y:6464.035 Z:31.627

Police Station X:436.491 Y: -982.172 Z:30.699

Humane Labs Entrance X:3619.749 Y:2742.740 Z:28.690

Burnt FIB Building X:160.868 Y:-745.831 Z:250.063

10 Car Garage Back Room X:223.193 Y:-967.322 Z:99.000

Humane Labs Tunnel X:3525.495 Y:3705.301 Z:20.992

Ammunation Office X:12.494 Y:-1110.130 Z: 29.797

Ammunation Gun Range X: 22.153 Y:-1072.854 Z:29.797

Trevor's Meth Lab X:1391.773 Y:3608.716 Z:38.942

Pacific Standard Bank Vault X:255.851 Y: 217.030 Z:101.683

Lester's House X:1273.898 Y:-1719.304 Z:54.771

Floyd's Apartment X:-1150.703 Y:-1520.713 Z:10.633

FIB Top Floor X:135.733 Y:-749.216 Z:258.152

IAA Office X:117.220 Y:-620.938 Z:206.047

Pacific Standard Bank X:235.046 Y:216.434 Z:106.287

Fort Zancudo ATC entrance X:-2344.373 Y:3267.498 Z:32.811

Fort Zancudo ATC top floor X:-2358.132 Y:3249.754 Z:101.451

Torture Room X: 147.170 Y:-2201.804 Z:4.688

OUTDOOR LOCATIONS

Main LS Customs X:-365.425 Y:-131.809 Z:37.873

Very High Up X:-129.964 Y:8130.873 Z:6705.307

IAA Roof X:134.085 Y:-637.859 Z:262.851

FIB Roof X:150.126 Y:-754.591 Z:262.865

Maze Bank Roof X:-75.015 Y:-818.215 Z:326.176

Top of the Mt Chilad X:450.718 Y:5566.614 Z:806.183

Most Northerly Point X:24.775 Y:7644.102 Z:19.055

Vinewood Bowl Stage X:686.245 Y:577.950 Z:130.461

Sisyphus Theater Stage X:205.316 Y:1167.378 Z:227.005

Galileo Observatory Roof X:-438.804 Y:1076.097 Z:352.411

Kortz Center X:-2243.810 Y:264.048 Z:174.615

Chumash Historic Family Pier X:-3426.683 Y:967.738 Z:8.347

Paleto Bay Pier X:-275.522 Y:6635.835 Z:7.425

God's thumb X:-1006.402 Y:6272.383 Z:1.503

Calafia Train Bridge X:-517.869 Y:4425.284 Z:89.795

Altruist Cult Camp X:-1170.841 Y:4926.646 Z:224.295

Maze Bank Arena Roof X:-324.300 Y:-1968.545 Z:67.002

Marlowe Vineyards X:-1868.971 Y:2095.674 Z:139.115

Hippy Camp X:2476.712 Y:3789.645 Z:41.226

Devin Weston's House X:-2639.872 Y:1866.812 Z:160.135

Abandon Mine X:-595.342 Y: 2086.008 Z:131.412

Weed Farm X:2208.777 Y:5578.235 Z:53.735

Stab City X: 126.975 Y:3714.419 Z:46.827

Airplane Graveyard Airplane Tail X:2395.096 Y:3049.616 Z:60.053

Satellite Dish Antenna X:2034.988 Y:2953.105 Z:74.602

Satellite Dishes X: 2062.123 Y:2942.055 Z:47.431

Windmill Top X:2026.677 Y:1842.684 Z:133.313

Sandy Shores Building Site Crane X:1051.209 Y:2280.452 Z:89.727

Rebel Radio X:736.153 Y:2583.143 Z:79.634

Quarry X:2954.196 Y:2783.410 Z:41.004

Palmer-Taylor Power Station Chimney X: 2732.931 Y: 1577.540 Z:83.671

Merryweather Dock X: 486.417 Y:-3339.692 Z:6.070

Cargo Ship X:899.678 Y:-2882.191 Z:19.013

Del Perro Pier X:-1850.127 Y:-1231.751 Z:13.017

Play Boy Mansion X:-1475.234 Y:167.088Z:55.841

Jolene Cranley-Evans Ghost X:3059.620 Y:5564.246 Z:197.091

NOOSE Headquarters X:2535.243 Y:-383.799 Z:92.993

Snowman X: 971.245 Y:-1620.993 Z:30.111

Oriental Theater X:293.089 Y:180.466 Z:104.301

Beach Skatepark X:-1374.881 Y:-1398.835 Z:6.141

Underpass Skatepark X:718.341 Y:-1218.714 Z: 26.014

Casino X:925.329 Y:46.152 Z:80.908

University of San Andreas X:-1696.866 Y:142.747 Z:64.372

```
La Puerta Freeway Bridge X: -543.932 Y:-2225.543 Z:122.366

Land Act Dam X: 1660.369 Y:-12.013 Z:170.020

Mount Gordo X: 2877.633 Y:5911.078 Z:369.624

Little Seoul X:-889.655 Y:-853.499 Z:20.566

Epsilon Building X:-695.025 Y:82.955 Z:55.855 Z:55.855

The Richman Hotel X:-1330.911 Y:340.871 Z:64.078

Vinewood sign X:711.362 Y:1198.134 Z:348.526

Los Santos Golf Club X:-1336.715 Y:59.051 Z:55.246

Chicken X:-31.010 Y:6316.830 Z:40.083

Little Portola X:-635.463 Y:-242.402 Z:38.175

Pacific Bluffs Country Club X:-3022.222 Y:39.968 Z:13.611

Vinewood Cemetery X:-1659993 Y:-128.399 Z:59.954

Paleto Forest Sawmill Chimney X:-549.467 Y:5308.221 Z:114.146

Mirror Park X:1070.206 Y:-711.958 Z:58.483

Rocket X:1608.698 Y:6438.096 Z:37.637

El Gordo Lighthouse X:3430.155 Y:5174.196 Z:41.280
```

## Content Replication Assignment

Teleport the player to a beach, spawn ten whales on the shore and generate an NPC wandering aroud them and take a screenshot in the style of HAPP V2.

# Chapter 7

# Hyperrealism

//intro to the artistic current of photorealism and hyperrealism in painting and drawing, connected to the idea of photorealism and simulation in games (attempt to simulate life itself, not just photography), relationship between the game and the physical world, the way virtual spaces influence and shape society (training self driving cars in GTA V, CGI shaping architecture of buildings and object…), the blurrying of the lines between virtual and physical…

### *8k* by Aram Bartholl

Aram Bartholl, *8k*, installation view

Aram Bartholl, *8k*, installation view

More about 8k

### Getting there

- Land Act Dam, Tataviam Mountains

## Readings

## Tutorial

### Setting Camera Views

GTA V has 4 default camera views, which can be switched by pressing the `V` key on PC. To set a specific camera view we can use the native function

SET_FOLLOW_PED_CAM_VIEW_MOD, followed by number 0, 1, 2 or 4 to establish the desired point of view:

- 0 - Third Person View - Close
- 1 - Third Person View - Mid
- 2 - Third Person View - Far
- 4 - First Person View

Switch to first person view:

```
Function.Call(Hash.SET_FOLLOW_PED_CAM_VIEW_MODE, 4);
```

## Setting the Time of the Day

We can control the time and the light for our shots to the second in GTA V. We can specify the exact time of the day with the native function SET_CLOCK_TIME, followed by hour, minute and second. Note that the sun sets at 5:30 a.m. and goes down at 8 p.m.

Set the imte of the day to 15:45:

```
Function.Call(Hash.SET_CLOCK_TIME, 15, 45, 00);
```

## Controlling the Weather

We can choose among the following weather options using the native function SET_WEATHER_TYPE_NOW_PERSIST:

“CLEAR” “EXTRASUNNY” “CLOUDS” “OVERCAST” “RAIN” “CLEARING” “THUNDER” “SMOG” “FOGGY” “XMAS” “SNOWLIGHT” “BLIZZARD”

Set the weather to blizzard:

```
Function.Call(Hash.SET_WEATHER_TYPE_NOW_PERSIST, "BLIZZARD");
```

## Scripting Cinematic Fade Out / In

Script Hook has native functions to create a fead to/from black. The function hash are DO_SCREEN_FADE_OUT and DO_SCREEN_FADE_IN and they are followed by the number of milliseconds to go from full black to showing the scene and viceversa.

Let’s create a fade to black over 3 seconds when we press the letter key ‘O’:

```
if (e.KeyCode == Keys.O)
{
    Function.Call(Hash.DO_SCREEN_FADE_OUT, 3000);
}
```

And a fade over 3 seconds when we press the letter key 'I':

```
if (e.KeyCode == Keys.I)
{
    Function.Call(Hash.DO_SCREEN_FADE_IN, 3000);
}
```

We could also create an automated check in our onTick loop, which keeps seeing if the screen has been faded to black. We can use the native function `IS_SCREEN_FADED_OUT` which is a boolean data type. This means it will return either true or false. If it returns true, it means the screen has been faded out.

Let's add an "if" statement in our onTick loop to chek if the screen has been faded to black, and if so we teleport somewhere else and we call a fade in over 3 second:

```
if (Function.Call<bool>(Hash.IS_SCREEN_FADED_OUT))
{
    Function.Call(Hash.SET_ENTITY_COORDS, Game.Player.Character, -1374.881f, -1398.835f, 6.141f,
    Wait(500);
    GTA.Native.Function.Call(Hash.DO_SCREEN_FADE_IN, 3000);
}
```

Now if you try to hit the 'O' key, the screen will fade out, and then it will automatically fade in again.

## Natural Vision Evolved Mod

Natural Vision Evolved (NVE) is a graphic mod develped by Jamal Rashid, aka Razed. This mod enhances GTA V's lighting, weather effects, ambient colours, world textures, building models, pushing the photo-realism and cinematic looks. While the mod contains settings for different hardware settings, it's recommended to have a relatively powerful PC with a good graphic card. Here you can find minimum and recommended requirements for Natural Vision Evolved mod.

Installation and setup:

- Go to razedmods.com/gta-v and download Natural Vision Evolved (6.2 Gb).

- Go to openiv.com/ and download Open IV, Open 'ovisetup' and install Open IV on your computer.
- Select GTA V Windows. Choose Grand Theft Auto V folder `C:\Program Files (x86)\Steam\steamapps\common\Grand Theft Auto V`
- Once Open IV is open, go to your file window select the `Tools` menu on top of the window, and select `ASI Manager`. In `ASI Manager` install all options: ASI Loader, OpenIV.ASI and openCamera.
- Select `Tools` again and click `Options`. Click on the `"mods" folder` tab and select `Allow edit mode only for archive inside "mods" folder`. Click `Close`.
- Select `Edit mode` at the top right of the window. Select `OK` on the pop up window.
- Now you can add your mod to the mods folder.
- Open the NVE mod folder, extract and select NaturalVision Installer PART ONE and drag it to Open IV. Install the file and select "mod folder" when asked to choose. After that is complete, extract and select NaturalVision Installer PART TWO and drag it to Open IV. Install the file and select "mod folder" when asked to choose. The Installers have to be executed in order: first do PART ONE, and then do PART TWO.
- Go back to your downloads and inside the NVE folder you can choose some optional addons. Install/Uninstall them with Open IV as above. Always choose to select "mod" folder and select install.
- Open GTA V, press `ESC` to bring up the menu and go to `SETTINGS`. Adjust the graphics quality, making sure `Shader Quality`, `Particle Quality` and `Post FX`are set to `<Very High>`. Restart the game to make change in effect. More details about installation are provided in the README file inside the NVE zip folder.

## ReShade

ReShade is a generic post-processing injector for games and video software developed by crosire.

Installation and setup:

- Go to reshade.me and download the latest version
- Open ReShade Setup and select Browse and navigate to your Grand Theft Auto V folder and select GTA 5.exe. Leave Direct X 10/11/12 checked and click Next. When asked to select presets to install click on Skip.

- Select SweetFX by CeeJay.dk and all the effect packages you want to install (OtisFX by Otis_Inf and CobraFX by SirCobra are particularly interesting for simulation of analogue photography, including depth of field) and complete installation.
- Go to GTA 5 Redux page and download the latest GTA5_REDUX.zip folder. This contains different presets to change the appearance of the game. Note that the file is more than 3 GB.
- Unzip the conent of the .zip folder and go to GTA_5_REDUX_RESHADE folder. Select all the files and copy them. Go to your Grand Theft Auto V folder > reshade-shaders, and create a Presets folder. Paste the preset files from GTA_5_REDUX_RESHADE in the Presets folder.
- Run GTA V and press the `Home key` to bring up the menu. The first time you bring up Reshade, it will offer to give a short tutorial on how to choose and modify the presets.
- Bring up a preset and mess around with its setting to see how it affects the visuals of the game. Thanks to different effects and presets, the aesthetics of the game world can be configured to obtain looks that are completely different from the original graphics created by the deleopers and designers of the game.

## Content Replication Assignment

# Chapter 8

# Glitch Art

//intro to error in photography (Clément Chéroux) and glitch art

### *Captures* by Raphael Brunk

Raphael Brunk, *Capture75011.12_23* , 2016

Raphael Brunk, *Capture55326.4_19*, 2016

More about Captures

### Getting there

- 3668 Wild Oats Dr, Vinewood Hills

## Readings

## Tutorial

### Controlling the Game Camera

We can detach the camera from the player's character and move freely around the game world. We instantiate a new `Camera` variable called `FreeCam`. We define it as the new world camera with the same position and rotation of the gameplay camera:

```
FreeCam = World.CreateCamera(GameplayCamera.Position, GameplayCamera.Rotation, GameplayCamera.Fie
```

We then define our new camera as the main camera with `World.RenderingCamera = FreeCam`. In order to control the camera we basically its vectors in 3D coordinates and multiply with a number over a certain direction to move towards a specific destination. A free camera – unlike the main character view – also allows us to move through the game architecture and terrain, revealing the construction of the game world. Press `O` to toggle the free camera On and off.

Example code

```
//this code was kindly provided by LeeC22 on https://gtaforums.com/topic/981454-free-cam-mode-in-

using System;
using System.Windows.Forms;
using GTA;
using GTA.Math;

using Control = GTA.Control;

namespace BasicFreeCamTest
{
    public class cBasicFreeCamTest : Script
    {
        private Keys ActivationKey = Keys.O;
        private bool FreeCamActive = false;
        private Camera FreeCam;

        public cBasicFreeCamTest()
        {
            Tick += onTick;
            KeyUp += onKeyUp;
            Aborted += onAborted;

            Interval = 0;
        }

        private void onTick(object sender, EventArgs e)
        {
            // Exits from the loop if the game is loading
            if (Game.IsLoading) return;

            if (FreeCamActive) UpdateFreeCam();
        }
```

```
        private void UpdateFreeCam()
        {
            float deltaTime = Game.LastFrameTime;
            float speed = 5f;
            float camSpeed = speed * deltaTime;
            float rotSpeed = 40f;
            float camRotSpeed = rotSpeed * deltaTime;

            Game.DisableAllControlsThisFrame(2);

            Vector3 camForward = FreeCam.Direction.Normalized; //FreeCam.ForwardVector
            Vector3 camRight = Vector3.Cross(Vector3.WorldUp, camForward); //FreeCam.R
            Vector3 camUp = Vector3.Cross(camRight, camForward); //FreeCam.UpVector?

                if (Game.IsDisabledControlJustPressed(2, Control.FrontendCancel))
            {
                World.RenderingCamera = null;
                FreeCam.Destroy();
                FreeCamActive = false;
                return;
            }

            float fbSpeedMult = Game.GetDisabledControlNormal(2, Control.MoveUpDown);
            float lrSpeedMult = Game.GetDisabledControlNormal(2, Control.MoveLeftRight)
            float yawSpeedMult = Game.GetDisabledControlNormal(2, Control.LookLeftRight
            float pitchSpeedMult = Game.GetDisabledControlNormal(2, Control.LookUpDown)

            float fbSpeed = camSpeed * fbSpeedMult;
            float lrSpeed = camSpeed * lrSpeedMult;
            float yawSpeed = camRotSpeed * yawSpeedMult;
            float pitchSpeed = camRotSpeed * pitchSpeedMult;

            Vector3 camLR = camRight * -lrSpeed;
            Vector3 camFB = camForward * -fbSpeed;
            Vector3 camMove = camLR + camFB;

            Vector3 camRot = new Vector3(pitchSpeed, 0, -yawSpeed);

            FreeCam.Position += camMove;
            FreeCam.Rotation += camRot;
        }

        private void onKeyUp(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == ActivationKey)
```

```
            {
                //toggle the freecam on/off
                FreeCamActive = !FreeCamActive;

                if (FreeCamActive) //ON
                {
                    FreeCam = World.CreateCamera(GameplayCamera.Position, GameplayCamera.Rotation
                    World.RenderingCamera = FreeCam;
                    UI.ShowSubtitle("FreeCam ON", 1000);
                }
                else //OFF
                {
                    World.RenderingCamera = null;
                    FreeCam.Destroy();
                    UI.ShowSubtitle("FreeCam OFF", 1000);
                }
            }
        }

        private void onAborted(object sender, EventArgs e)
        {
            World.RenderingCamera = null;
        }
    }
}
```

## Recording and Replaying Camera Movements

We can record the movement of the camera by storing its posiiton and movement every frame in a list of Vector3 variables. Then we can simply access the list and assign the position of the camera at each frame. This is useful to create reusable camera movements, like tracking shots and cinematic camera movements. Once we have achieved the desired camera movement, we can spawn objects in the scene and adjust color grading before saving the screengrab. In this example, turn on the free camera by pressing `O` and then start and stop the recording of the camera movement by pressing `I`. To toggle the replay camera press `K`.

Example code

```
//this code was extended by code for controlling the camera by LeeC22 on https://gtaforums.com/to
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;

using System.Windows.Forms;
using System.Drawing;
using GTA;
using GTA.Math;
using GTA.Native;


using Control = GTA.Control;

namespace BasicFreeCamTest
{
    public class cBasicFreeCamTest : Script
    {
        private Keys ActivationKey = Keys.O;
        private Keys RecordKey = Keys.I;
        private Keys ReplayKey = Keys.K;
        private bool FreeCamActive = false;
        private bool RecCamActive = false;
        private int CamFramesIndex = 0;
        private bool ReplayCamActive = false;
        private Camera FreeCam;
        private Camera ReplayCam;
        List<Vector3> RecCamPos = new List<Vector3>();
        List<Vector3> RecCamRot = new List<Vector3>();

        public cBasicFreeCamTest()
        {
            Tick += onTick;
            KeyUp += onKeyUp;
            Aborted += onAborted;

            Interval = 0;
        }

        private void onTick(object sender, EventArgs e)
        {
            // Exits from the loop if the game is loading
            if (Game.IsLoading) return;

            if (FreeCamActive) UpdateFreeCam();

            if (RecCamActive) UpdateRecCam();
```

```
            if (ReplayCamActive) UpdateReplayCam();
        }

        private void UpdateRecCam()
        {
            RecCamPos.Add(FreeCam.Position);
            RecCamRot.Add(FreeCam.Rotation);
            UI.ShowSubtitle("Recording Camera ON", 100);
        }
        private void UpdateReplayCam()
        {

            Game.DisableAllControlsThisFrame(2);

            ReplayCam.Position = RecCamPos[CamFramesIndex];
            ReplayCam.Rotation = RecCamRot[CamFramesIndex];

            CamFramesIndex++;
            if (CamFramesIndex >= RecCamPos.Count)
            {
                CamFramesIndex = RecCamPos.Count - 1;
                ReplayCamActive = false;
            }
            else
            {
                UI.ShowSubtitle("Playing Back Camera\nPlaying Back Frame #" + CamFramesIndex, 100
            }
        }

        private void UpdateFreeCam()
        {
            float deltaTime = Game.LastFrameTime;
            float speed = 2f;
            float camSpeed = speed * deltaTime;
            float rotSpeed = 80f;
            float camRotSpeed = rotSpeed * deltaTime;

            Game.DisableAllControlsThisFrame(2);

            Vector3 camForward = FreeCam.Direction.Normalized;
            Vector3 camRight = Vector3.Cross(Vector3.WorldUp, camForward);
            Vector3 camUp = Vector3.Cross(camRight, camForward);

            if (Game.IsDisabledControlJustPressed(2, Control.FrontendCancel))
            {
```

```
            World.RenderingCamera = null;
            FreeCam.Destroy();
            FreeCamActive = false;
            return;
        }

        float fbSpeedMult = Game.GetDisabledControlNormal(2, Control.MoveUpDown);
        float lrSpeedMult = Game.GetDisabledControlNormal(2, Control.MoveLeftRight)
        float yawSpeedMult = Game.GetDisabledControlNormal(2, Control.LookLeftRight
        float pitchSpeedMult = Game.GetDisabledControlNormal(2, Control.LookUpDown)

        float fbSpeed = camSpeed * fbSpeedMult;
        float lrSpeed = camSpeed * lrSpeedMult;
        float yawSpeed = camRotSpeed * yawSpeedMult;
        float pitchSpeed = camRotSpeed * pitchSpeedMult;

        Vector3 camLR = camRight * -lrSpeed;
        Vector3 camFB = camForward * -fbSpeed;
        Vector3 camMove = camLR + camFB;

        Vector3 camRot = new Vector3(pitchSpeed, 0, -yawSpeed);

        FreeCam.Position += camMove;
        FreeCam.Rotation += camRot;
    }

    private void onKeyUp(object sender, KeyEventArgs e)
    {
        if (e.KeyCode == ActivationKey)
        {
            //toggle the freecam on/off
            FreeCamActive = !FreeCamActive;

            if (FreeCamActive) //ON
            {
                FreeCam = World.CreateCamera(GameplayCamera.Position, GameplayCamer
                World.RenderingCamera = FreeCam;
                UI.ShowSubtitle("Free Camera ON", 1000);
            }
            else //OFF
            {
                World.RenderingCamera = null;
                FreeCam.Destroy();
                UI.ShowSubtitle("Free Camera OFF", 1000);
            }
```

```
        }
        if (e.KeyCode == RecordKey)
        {
            RecCamActive = !RecCamActive;
            if(RecCamActive)
            {
                //Clear the array of positions and rotations
                RecCamPos.Clear();
                RecCamRot.Clear();
                //reset index
                CamFramesIndex= 0;
                //disable replay cam
                ReplayCamActive= false;
                //activate free cam if it's off
                if (!FreeCamActive)
                {
                    FreeCam = World.CreateCamera(GameplayCamera.Position, GameplayCamera.Rota
                    World.RenderingCamera = FreeCam;
                    FreeCamActive = true;

                }
            }
            else
            {
                UI.ShowSubtitle("Recording Camera OFF\nRecorded Frames = " + RecCamPos.Count,
            }
        }
        if (e.KeyCode == ReplayKey)
        {
            ReplayCamActive = !ReplayCamActive;

            if(ReplayCamActive)
            {
                CamFramesIndex = 0;
                ReplayCam = World.CreateCamera(RecCamPos[0], RecCamRot[0], GameplayCamera.Fie
                RecCamActive = false;
                FreeCamActive = false;
            }
            else
            {
                RecCamActive = false;
                CamFramesIndex = 0;
            }
        }
    }
```

```
        private void onAborted(object sender, EventArgs e)
        {
            World.RenderingCamera = null;
        }
    }
}
```

### Attaching a Camera to an Entity

We can also attach a camera to a car, a prop or an NPC. The Camera object in `GTA` has `AttachTo` function we can call, followed by two paramenters: the entity we want to attach the camera to and the position relative to the entity. Attaching a camera to an NPC can provide interesting views and perspectives. Here we create a cat NPC by pressing `G` and attach a camera to it. We can switch the camera view by pressing `SHIFT + K` and let the cat follow the player by pressing `H`.

Example code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using GTA;
using GTA.Math;
using System.Windows.Forms;
using System.Drawing;
using GTA.Native;
using System.IO;


namespace moddingTutorial
{
    public class moddingTutorial : Script
    {
        Vector3 myCamPos;
        int CamSelect = 0;
        Ped newPed = null;
        Camera myCam;

        public moddingTutorial()
```

```
        {
            this.Tick += onTick;
            this.KeyUp += onKeyUp;
            this.KeyDown += onKeyDown;
        }

        private void onTick(object sender, EventArgs e) //this function gets executed continuousl
        {
            //exits from the loop if the game is loading
            if (Game.IsLoading) return;

            //update the cameras if the ped is spawn
            if (newPed != null)
            {
                //create the cameras if none have been created yet.
                if (myCam == null)
                {
                    UI.ShowSubtitle("Set new camera");
                    myCam = World.CreateCamera(Vector3.Zero, newPed.Rotation, 50f);
                    // Set the camera position (relative pos)
                    myCamPos = new Vector3(0, 0, 1f);
                }
                //attach the cameras
                myCam.AttachTo(newPed, myCamPos);
                //sync rotation
                myCam.Rotation = newPed.Rotation;
            }
        }


        private void onKeyUp(object sender, KeyEventArgs e)//everything inside here is executed c
        {
            //press control+K to switch between gameplay default camera and the NPC camera
            if (e.KeyCode == Keys.K && e.Modifiers == Keys.Shift && newPed != null)
            {
                CamSelect = (CamSelect + 1) % 2;
                switch (CamSelect)
                {
                    case 0: World.RenderingCamera = null;
                        UI.ShowSubtitle("Showing Gameplay Cam View");
                        break;
                    case 1: World.RenderingCamera = myCam;
                        UI.ShowSubtitle("Showing NPC Cam View");
                        break;
                }
```

```
            }
        }

        private void onKeyDown(object sender, KeyEventArgs e) //everything inside here
        {
            if(e.KeyCode == Keys.G)
            {
                //spawn new Ped
                newPed = World.CreatePed(PedHash.Cat, Game.Player.Character.GetOffsetIn
            }

            if (e.KeyCode == Keys.H)
            {
                //follow player (persistent)
                Function.Call(Hash.TASK_FOLLOW_TO_OFFSET_OF_ENTITY, newPed.Handle, Game
                //look at player
                newPed.Task.LookAt(Game.Player.Character);
            }
            if (e.KeyCode == Keys.J)
            {
                //stop NPC
                newPed.Task.ClearAll();
            }
            if (e.KeyCode == Keys.L)
            {
                //delete ped
                newPed.Delete();
            }
        }
    }
}
```

## Creating and Switching between Multiple Cameras

The easiest way to create multiple cameras is position invisible Ped NPCs in specific locations, make them invisible with the native function `SET_ENTITY_VISIBLE` and attach a camera to them. We can switch between the different cameras with `SHIFT + K`. These can be used as different cameras to surveil and look at scenes in different locations within the game world.

Example code

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using GTA;
using GTA.Math;
using System.Windows.Forms;
using System.Drawing;
using GTA.Native;

namespace ModdingTutorial
{
    public class ModdingTutorial : Script
    {
        Vector3 myCamPos;
        int CamSelect = 0;
        int CamCount = 3;

        List<Ped> myPeds = new List<Ped>();
        Model myModel = PedHash.Abigail;
        List<Vector3> myLocs = new List<Vector3>();
        Camera newCam= null;
        List<Camera> myCam = new List<Camera>();


        public ModdingTutorial()
        {
            this.Tick += onTick;
            this.KeyUp += onKeyUp;
            this.KeyDown += onKeyDown;

            myLocs.Add(new Vector3(450.178f, 5566.614f, 806.183f)); //Mt.Chiliad
            myLocs.Add(new Vector3(24.775f, 7644.102f, 18.055f)); //Most Northern Point
            myLocs.Add(new Vector3(150.126f, -754.591f, 261.865f)); //FIB Roof

            for (int i = 0; i < CamCount; i++)
            {
                //create Ped
                var newPed = World.CreatePed(myModel, myLocs[i]);
                Function.Call(Hash.SET_ENTITY_VISIBLE, newPed, false, 0);
                myPeds.Add(newPed);

                //create Cam
                newCam = World.CreateCamera(Vector3.Zero, myPeds[i].Rotation, 50f);
                myCam.Add(newCam);
                myCamPos = new Vector3(0, 0, 1f);
```

```
            }

        }

        private void onTick(object sender, EventArgs e)
        {
            //update Cams
            for (int i = 0; i < CamCount; i++)
            {
                myCam[i].AttachTo(myPeds[i], myCamPos);
                myCam[i].Rotation = myPeds[i].Rotation;
            }
        }

        private void onKeyUp(object sender, KeyEventArgs e)
        {
            //switch between cameras when pressing SHIFT + K
            if(e.KeyCode == Keys.K && e.Modifiers == Keys.Shift && myPeds[2] != null)
            {
                CamSelect = (CamSelect + 1) % 4;
                switch(CamSelect)
                {
                    case 0: World.RenderingCamera = null;
                        UI.ShowSubtitle("Showing Gameplay Cam View");
                        break;
                    case 1: World.RenderingCamera = myCam[0];
                        UI.ShowSubtitle("Showing Cam 1");
                        break;
                    case 2: World.RenderingCamera = myCam[1];
                        UI.ShowSubtitle("Showing Cam 2");
                        break;
                    case 3: World.RenderingCamera = myCam[2];
                        UI.ShowSubtitle("Showing Cam 3");
                        break;
                }
            }
        }

        private void onKeyDown(object sender, KeyEventArgs e)
        {
        }
    }
}
```

## Switching Character through Satellite Camera View

We can use the native function `START_PLAYER_SWITCH` to enable the satellite view animation from the game, and get transported to a different location where an NPC or player avatar is spawned. Press `G` to move across divverent NPCs at different location through the satellite camera transition.

Example code

```
/*
    this was adapted from code shared by LeeC22 on gtaforums.com
    https://gtaforums.com/topic/951002-c-looking-for-player-switch-sample-solved-by-me/#comment-1
*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using GTA;
using GTA.Math;
using System.Windows.Forms;
using System.Drawing;
using GTA.Native;
using System.IO;


namespace moddingTutorial
{
    public class moddingTutorial : Script
    {
        Ped newPed = null;
        Vector3 SwitchLocation2;
        List<Vector3> switchLocations = new List<Vector3>();
        int index = 0;
        List<String> models = new List<String>();
        int modelIndex = 0;

        public moddingTutorial()
        {
            this.Tick += onTick;
            this.KeyUp += onKeyUp;
            this.KeyDown += onKeyDown;

        //add locations to the switchLocations list
```

```
            switchLocations.Add(new Vector3(24.775f, 7644.102f, 18.055f)); //Most Nort
            switchLocations.Add(new Vector3(-595.342f, 2086.008f, 130.412f)); //Mine
            switchLocations.Add(new Vector3(150.126f, -754.591f, 261.865f)); //FIB Roo

        //add models to the models list
            models.Add("s_m_m_doctor_01");
            models.Add("s_m_m_migrant_01");
            models.Add("a_c_cormorant");
            models.Add("a_c_deer");
            models.Add("a_c_pug");
        }

        private void onTick(object sender, EventArgs e) //this function gets executed
        {

            //If the character switch is in process
            if (Function.Call<bool>(Hash.IS_PLAYER_SWITCH_IN_PROGRESS))
            {
                //If Switch State is 8 - that's the point when it starts dropping to t
                if (Function.Call<int>(Hash.GET_PLAYER_SWITCH_STATE) == 8)
                {
                    //Set the player to the switch location
                    Game.Player.Character.Position = switchLocations[index];

                    //Generate the hash for the chosen model
                    int poshHash = Game.GenerateHash(models[modelIndex]);

                    //Create the model
                    Model poshModel = new Model(poshHash);

                    //Check if it is valid
                    if (poshModel.IsValid)
                    {
                        //Wait for it to load, should be okay because it was used to c
                        while (!poshModel.IsLoaded)
                        {
                            Wait(100);
                        }

                        //Change the player model to the target ped model
                        Function.Call(Hash.SET_PLAYER_MODEL, Game.Player, poshHash);

                        //Let the game clean up the created Model
                        poshModel.MarkAsNoLongerNeeded();
                    }
```

```
                else
                {
                    //Falls to here if the model valid check fails
                    Function.Call(Hash.SET_PLAYER_MODEL, Game.Player, (int)PedHash.Tourist01A
                }

                //Delete the target ped as it's no longer needed
                newPed.Delete();


                // Set the switch outro based on the gameplay camera position
                // Function.Call((Hash)0xC208B673CE446B61, camPos.X, camPos.Y, camPos.Z, camR

                Function.Call((Hash)0xC208B673CE446B61, GameplayCamera.Position.X, GameplayCa

                //Call this unknown native that seems to finish things off
                Function.Call(Hash._0x74DE2E8739086740);

            //Make the character wander around autonomously
                    Game.Player.Character.Task.WanderAround();
                }
            }

        }


        private void onKeyUp(object sender, KeyEventArgs e)
        {

        }

        private void onKeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.G)
            {
        //Stop previous tasks
                Game.Player.Character.Task.ClearAll();

                //Move the index to the next location
                index++;
                if (index >= switchLocations.Count) index = 0;

        //Move the index to the new ped model
                modelIndex++;
                if (modelIndex >= models.Count) modelIndex = 0;
```

```
                //Create the ped to switch to
                newPed = World.CreatePed(models[modelIndex], switchLocations[index]);

                //Native function to initiate the switch Function.Call(Hash.START_PLAY
                Function.Call(Hash.START_PLAYER_SWITCH, Game.Player.Character.Handle, 

            }
        }
    }
}
```

## Menyoo Mod Trainer

Game trainers are a kind of game modification that changes its behavior using addresses and values. Trainers are used to gain unfair advantage in games and cheating, but they are also used by players to "train" themselves under different game conditions. In GTA V a trainer mod is very useful for creative practices, as it allows players to modify many aspects of the game that extend the possibilities offered by the official Scene Director and Rockstar Editor. They do not require scripting and offer control of the game world through a more intuitive graphic interface.

Menyoo is one of many available mod trainers for GTA V. It's the most popular trainer because of its incredibly wide range of features, controlling NPCs, vehicles, props and scenes, and allowing players to generate custom interiors and objects from GTA V's database of entities, dynamic scenarios, and entire sets and scenes for machinima and photographic projects.

Installation and setup:

Menyoo requires Scripthook V and Scripthook V Dot Net which we already installed in order to enable our scripts (check chapter 5 if for some reasons you do not have them installed).

- In order to install the Menyoo Single Player Trainer go to github.com/MAFINS/MenyooSP/releases and download the `MenyooSP.zip` file in the Assets section of the page. Open its content and select the `menyooStuff``folder and`Menyoo.asi"'file. Copy and paste these in your GTA V directory.
- Run GTA V and press `F8` to bring up the Menyoo Trainer menu. From there you can control many of the functions we are learning to script, including character models, animations, and teleporting.

//https://forums.gta5-mods.com/topic/64/menyoo-object-spooner-tutorial

Scripting will always give you more accurate control of what you can achieve, but a trainer mod is very useful in showing what's possible and it's a much more accessible tool for those who are less incline to code. Finally, all of the tools in this guide are not mutually exclusive, but can be combined together: menyoo can be used with custom scripts, ReShade, screenshotting, Scene Director and Rockstar Editor, allowing the player to really have complete control of the game and its world. In the example below we have used code to spawn 5 clown NPCs and to position the game camera, the weather manipulated in Menyoo to create a meteor shower, and we used a shallow depth of field lens and color grading in ReShade.

## CodeWalker

CodeWalker is an applicaiton created by dexyfex that renders an interactive 3D Map for GTAV. Through CodeWalker we are able to modify the architecture and objects inside the map, replacing textures and images with custom content.

Installation and setup:

- Go to https://www.gta5-mods.com/tools/codewalker-gtav-interactive-3d-map and download CodeWalker. Extract the content of the .zip folder and open the CodeWalker application. Once the GTAV installation folder has been found, the world view will load by default. Use the WASD keys to move, and left-drag to rotate the camera view. Mouse wheel zooms in/out and controls the movement speed.

- Click on the arrow on the top left of the screen, then select `Enable Mods` and `Enable DLC`.

- On top of the panel click on `Options > Lighting`. Deselect `Deferred shading` and in `Time of Day` set the time to 12:00. Then click on `Save Settings`.

- Next we are going to hit the `T` key and select the `Select objects` tool from the tool bar.

- Click on the `Move` tool, which allows to select things by right clicking on them. Hovering on the object will highlight the bounding box of the model, right clicking it will enable the movement of the object. Try dragging an object in 3D space, clicking and dragging one of the 3D axis arrows.

- Undo the changes by pressing `Undo` from the tool bar or `Ctrl + Z` until the object is back in its original position.

- Now press the first tool on the left in the tool bar: `New`. This will open the Project window.

- Select an object with the `Select objects` tools and you will see that the object information is updated in the Project window, including the object's position, rotation and file name (archetype). Choose an object and click `Add to Project`. This will add the object to our ymap.

Note: do not delete assets from the game map. Simply select an object that can be added to the project.

- With the `Select objects` tool and `Move` tool selected, hold `SHIFT` and drag the object to multiply the object. In this example we have multiplied a chair a few times, moved them and rotated them.
- Once you are happy with your changes, go to `File > New > Ymap File` in the Project window. This creates a new ymap called `map1.ymap`.
- We can also add objects that are not in the immediate vicinity of the location we want to add them to. Select map1.ymap in the Project window, and from the window menu press `Ymap > New Entity`. This will create a default egg. In the Project window go to Archetype and replace the file name with the one of the object you want. You can refer to this database to find the file name reference of each object easily (or use the `Select objects` tools and copy the Archetype info from your desired object).
- Once you are done click on the `Save` icon and save your file in a folder titled YMAP.
- In the Project window now go to `Tools` and click on `Manifest Generator...`. Click on `Save _manifest.yml` and save them as _manifest1.yml //memo: actually this is only needed for custom YMap on FiveM, so we could skip it here
- Go to Open IV > mods > update > x64 > dlcpacks, click Edit Mode and create a folder named custom_maps. Inside the folder copy the file dlc.rpf that you can find here.
- Now go to Open IV > mods > update > update.rpf > common > data. Find dlclist.xml, richt click on it and choose edit. Scroll to the end of the file and add `<Item>dlcpacks:/custom_maps/</Item> above </Paths>` and save.
- Finally go to Open IV > mods > update > x64 > dlcpacks > custom_maps > dlc.rpf > x64 > levels > gta5 > _citye > maps > Custom_maps.rpf. Make sure Edit Mode is on and place your .ymap file here.
- Now open GTA V and go to the location where you made the changes and see if they are there.

- Depending on the kind of objects you have place, and the flags you set, you will be able to interact with them, and the game will also recognize them and span some NPCs on them.

This tutorial was adapted from [GTA 5 Tutorial - Part 2] Using codewalker and creating/modifying ymaps by Tobii and https://www.youtube.com/watch?v=70ODCkbrj60&ab_channel=TeachingHub

//https://www.gta5-mods.com/tools/codewalker-gtav-interactive-3d-map
//https://www.youtube.com/watch?v=VDoAjV3kv2g&list=PL3KgAhmoIEOU540c5hydM5KaVXeJcYiPF&ab_channel=Tobii

//delete objects (note: if Max LOD > LOD shows the object, you should not delete it, just move it but dont delete it)

//generate and save manifestNAME.ymf // copy ymf and ymap files to server

//replace textures with custom images

## Content Replication Assignment

# Chapter 9

# Meta-Photography

// General intro on the artistic reflection on the photographic medium itself, the tradition of conceptual photography and the connection to the simulation of the camera and the act of photographing virtual worlds

### GTA V Photography Bot

### Crossroad of Realities by Benoit Paillé

## Readings

## Tutorial

### Virtual Keys

The most reliable way to send keyboard and mouse input is the SendInput function in user32.dll.

The SendInput function takes three parameters: the number of inputs, an array of INPUT for the inputs we want to send, and the size of our INPUT struct. The INPUT struct includes an integer that indicates the type of input and a union for the inputs that will be passed.

Keys are mapped to Direct Input Keyboard hex codes. You can find a list of all hex codes for each key here or below.

List of DirectX key mappings

```
Value   Macro         Symbol
```

```
------------------------
0x01    DIK_ESCAPE  Esc
0x02    DIK_1         1
0x03    DIK_2         2
0x04    DIK_3         3
0x05    DIK_4         4
0x06    DIK_5         5
0x07    DIK_6         6
0x08    DIK_7         7
0x09    DIK_8         8
0x0A    DIK_9         9
0x0B    DIK_0         0
0x0C    DIK_MINUS     -
0x0D    DIK_EQUALS  =
0x0E    DIK_BACK      Back Space
0x0F    DIK_TAB     Tab
0x10    DIK_Q         Q
0x11    DIK_W         W
0x12    DIK_E         E
0x13    DIK_R         R
0x14    DIK_T         T
0x15    DIK_Y         Y
0x16    DIK_U         U
0x17    DIK_I         I
0x18    DIK_O         O
0x19    DIK_P         P
0x1A    DIK_LBRACKET  [
0x1B    DIK_RBRACKET  ]
0x1C    DIK_RETURN  Enter
0x1D    DIK_LContol Ctrl (Left)
0x1E    DIK_A         A
0x1F    DIK_S         S
0x20    DIK_D         D
0x21    DIK_F         F
0x22    DIK_G         G
0x23    DIK_H         H
0x24    DIK_J         J
0x25    DIK_K         K
0x26    DIK_L         L
0x27    DIK_SEMICOLON   ;
0x28    DIK_APOSTROPHE  '
0x29    DIK_GRAVE     `
0x2A    DIK_LSHIFT  Shift (Left)
0x2B    DIK_BACKSLASH   \
0x2C    DIK_Z         Z
0x2D    DIK_X         X
```

```
0x2E    DIK_C         C
0x2F    DIK_V         V
0x30    DIK_B         B
0x31    DIK_N         N
0x32    DIK_M         M
0x33    DIK_COMMA     ,
0x34    DIK_PERIOD  .
0x35    DIK_SLASH     /
0x36    DIK_RSHIFT  Shift (Right)
0x37    DIK_MULTIPLY    * (Numpad)
0x38    DIK_LMENU     Alt (Left)
0x39    DIK_SPACE     Space
0x3A    DIK_CAPITAL Caps Lock
0x3B    DIK_F1      F1
0x3C    DIK_F2      F2
0x3D    DIK_F3      F3
0x3E    DIK_F4      F4
0x3F    DIK_F5      F5
0x40    DIK_F6      F6
0x41    DIK_F7      F7
0x42    DIK_F8      F8
0x43    DIK_F9      F9
0x44    DIK_F10     F10
0x45    DIK_NUMLOCK Num Lock
0x46    DIK_SCROLL  Scroll Lock
0x47    DIK_NUMPAD7 7 (Numpad)
0x48    DIK_NUMPAD8 8 (Numpad)
0x49    DIK_NUMPAD9 9 (Numpad)
0x4A    DIK_SUBTRACT  - (Numpad)
0x4B    DIK_NUMPAD4 4 (Numpad)
0x4C    DIK_NUMPAD5 5 (Numpad)
0x4D    DIK_NUMPAD6 6 (Numpad)
0x4E    DIK_ADD     + (Numpad)
0x4F    DIK_NUMPAD1 1 (Numpad)
0x50    DIK_NUMPAD2 2 (Numpad)
0x51    DIK_NUMPAD3 3 (Numpad)
0x52    DIK_NUMPAD0 0 (Numpad)
0x53    DIK_DECIMAL . (Numpad)
0x57    DIK_F11     F11
0x58    DIK_F12     F12
0x64    DIK_F13     F13 (NEC PC-98)
0x65    DIK_F14     F14 (NEC PC-98)
0x66    DIK_F15     F15 (NEC PC-98)
0x70    DIK_KANA      Kana  Japenese Keyboard
0x79    DIK_CONVERT Convert Japenese Keyboard
0x7B    DIK_NOCONVERT   No Convert  Japenese Keyboard
```

```
0x7D    DIK_YEN     ¥     Japenese Keyboard
0x8D    DIK_NUMPADEQUALS    =   NEC PC-98
0x90    DIK_CIRCUMFLEX  ^   Japenese Keyboard
0x91    DIK_AT      @   NEC PC-98
0x92    DIK_COLON     : NEC PC-98
0x93    DIK_UNDERLINE   _   NEC PC-98
0x94    DIK_KANJI     Kanji Japenese Keyboard
0x95    DIK_STOP      Stop  NEC PC-98
0x96    DIK_AX      (Japan AX)
0x97    DIK_UNLABELED   (J3100)
0x9C    DIK_NUMPADENTER Enter (Numpad)
0x9D    DIK_RCONTROL    Ctrl (Right)
0xB3    DIK_NUMPADCOMMA , (Numpad)  NEC PC-98
0xB5    DIK_DIVIDE    / (Numpad)
0xB7    DIK_SYSRQ       Sys Rq
0xB8    DIK_RMENU       Alt (Right)
0xC5    DIK_PAUSE       Pause
0xC7    DIK_HOME        Home
0xC8    DIK_UP        ↑ (Arrow up)
0xC9    DIK_PRIOR       Page Up
0xCB    DIK_LEFT        ←   (Arrow left)
0xCD    DIK_RIGHT       →   (Arrow right)
0xCF    DIK_END       End
0xD0    DIK_DOWN        ↓   (Arrow down)
0xD1    DIK_NEXT        Page Down
0xD2    DIK_INSERT    Insert
0xD3    DIK_DELETE    Delete
0xDB    DIK_LWIN        Windows
0xDC    DIK_RWIN        Windows
0xDD    DIK_APPS        Menu
0xDE    DIK_POWER       Power
0xDF    DIK_SLEEP       Windows
```

Example code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using GTA;
using GTA.Math;
using System.Windows.Forms;
using System.Drawing;
using GTA.Native;
```

```
using System.IO;
using System.Runtime.InteropServices;

namespace moddingTutorial
{
    public class moddingTutorial : Script
    {

        //this bunch of stuff is to control keyboard and mouse
        [StructLayout(LayoutKind.Sequential)]
        public struct KeyboardInput
        {
            public ushort wVk;
            public ushort wScan;
            public uint dwFlags;
            public uint time;
            public IntPtr dwExtraInfo;
        }
        [StructLayout(LayoutKind.Sequential)]
        public struct MouseInput
        {
            public int dx;
            public int dy;
            public uint mouseData;
            public uint dwFlags;
            public uint time;
            public IntPtr dwExtraInfo;
        }
        [StructLayout(LayoutKind.Sequential)]
        public struct HardwareInput
        {
            public uint uMsg;
            public ushort wParamL;
            public ushort wParamH;
        }
        [StructLayout(LayoutKind.Explicit)]
        public struct InputUnion
        {
            [FieldOffset(0)] public MouseInput mi;
            [FieldOffset(0)] public KeyboardInput ki;
            [FieldOffset(0)] public HardwareInput hi;
        }
        public struct Input
        {
            public int type;
```

```
        public InputUnion u;
    }
    [Flags]
    public enum InputType
    {
        Mouse = 0,
        Keyboard = 1,
        Hardware = 2
    }
    [Flags]
    public enum KeyEventF
    {
        KeyDown = 0x0000,
        ExtendedKey = 0x0001,
        KeyUp = 0x0002,
        Unicode = 0x0004,
        Scancode = 0x0008
    }
    [Flags]
    public enum MouseEventF
    {
        Absolute = 0x8000,
        HWheel = 0x01000,
        Move = 0x0001,
        MoveNoCoalesce = 0x2000,
        LeftDown = 0x0002,
        LeftUp = 0x0004,
        RightDown = 0x0008,
        RightUp = 0x0010,
        MiddleDown = 0x0020,
        MiddleUp = 0x0040,
        VirtualDesk = 0x4000,
        Wheel = 0x0800,
        XDown = 0x0080,
        XUp = 0x0100
    }

    [DllImport("user32.dll", SetLastError = true)]
    private static extern uint SendInput(uint nInputs, Input[] pInputs, int cbSize);
    [DllImport("user32.dll")]
    private static extern IntPtr GetMessageExtraInfo();
    bool pressKey = false;


    public moddingTutorial()
```

```
        {
            this.Tick += onTick;
            this.KeyUp += onKeyUp;
            this.KeyDown += onKeyDown;
        }

        private void onTick(object sender, EventArgs e) //this function gets executed
        {
        }

        private void onKeyUp(object sender, KeyEventArgs e)
        {
        }

        private void onKeyDown(object sender, KeyEventArgs e)
        {

            if (e.KeyCode == Keys.H)
            {
                //define the pressing of the arrow up key
                Input[] keyUP_press = new Input[]
                {
                  new Input
                  {
                    type = (int)InputType.Keyboard,
                    u = new InputUnion
                    {
                      ki = new KeyboardInput
                      {
                        wVk = 0,
                        wScan = 0xC8, // key ARROW UP
                        dwFlags = (uint)(KeyEventF.KeyDown | KeyEventF.Scancode), //ke
                        dwExtraInfo = GetMessageExtraInfo()
                      }
                    }
                  }
                };

                //define the release of the arrow up key
                Input[] keyUP_release = new Input[]
                {
                  new Input
                  {
                    type = (int)InputType.Keyboard,
                    u = new InputUnion
```

```
                    {
                      ki = new KeyboardInput
                      {
                        wVk = 0,
                        wScan = 0xC8, //key ARROW UP
                        dwFlags = (uint)(KeyEventF.KeyUp | KeyEventF.Scancode), //key release
                        dwExtraInfo = GetMessageExtraInfo()
                      }
                    }
                  }
                };

                //we need to send a key press and add a delay before releasing it otherwise it's
                //send key press of arrow up key
                SendInput((uint)keyUP_press.Length, keyUP_press, Marshal.SizeOf(typeof(Input)));
                //delay half a second
                Wait(500);
                //send key release of arrow up key
                SendInput((uint)keyUP_release.Length, keyUP_release, Marshal.SizeOf(typeof(Input)
            }
        }
    }
}
```

GTA V Photographer Bot Script

```
//#define debug


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using GTA;
using GTA.Math;
using System.Windows.Forms;
using System.Drawing;
using GTA.Native;
using System.IO;
using System.Runtime.InteropServices;
```

```
namespace moddingTutorial
{
    public class moddingTutorial : Script
    {

        bool botOn = false; //turn the bot ON/OFF
        List<Vector3> switchLocations = new List<Vector3>(); //list of locations to te
        int index;

        //KEYBOARD AND MOUSE INPUT STUFF
        [StructLayout(LayoutKind.Sequential)]
        public struct KeyboardInput
        {
            public ushort wVk;
            public ushort wScan;
            public uint dwFlags;
            public uint time;
            public IntPtr dwExtraInfo;
        }
        [StructLayout(LayoutKind.Sequential)]
        public struct MouseInput
        {
            public int dx;
            public int dy;
            public uint mouseData;
            public uint dwFlags;
            public uint time;
            public IntPtr dwExtraInfo;
        }
        [StructLayout(LayoutKind.Sequential)]
        public struct HardwareInput
        {
            public uint uMsg;
            public ushort wParamL;
            public ushort wParamH;
        }
        [StructLayout(LayoutKind.Explicit)]
        public struct InputUnion
        {
            [FieldOffset(0)] public MouseInput mi;
            [FieldOffset(0)] public KeyboardInput ki;
            [FieldOffset(0)] public HardwareInput hi;
        }
        public struct Input
        {
```

```
        public int type;
        public InputUnion u;
    }
    [Flags]
    public enum InputType
    {
        Mouse = 0,
        Keyboard = 1,
        Hardware = 2
    }
    [Flags]
    public enum KeyEventF
    {
        KeyDown = 0x0000,
        ExtendedKey = 0x0001,
        KeyUp = 0x0002,
        Unicode = 0x0004,
        Scancode = 0x0008
    }
    [Flags]
    public enum MouseEventF
    {
        Absolute = 0x8000,
        HWheel = 0x01000,
        Move = 0x0001,
        MoveNoCoalesce = 0x2000,
        LeftDown = 0x0002,
        LeftUp = 0x0004,
        RightDown = 0x0008,
        RightUp = 0x0010,
        MiddleDown = 0x0020,
        MiddleUp = 0x0040,
        VirtualDesk = 0x4000,
        Wheel = 0x0800,
        XDown = 0x0080,
        XUp = 0x0100
    }


    [DllImport("user32.dll", SetLastError = true)]
    private static extern uint SendInput(uint nInputs, Input[] pInputs, int cbSize);
    [DllImport("user32.dll")]
    private static extern IntPtr GetMessageExtraInfo();
    bool pressKey = false;
```

```
        public moddingTutorial()
        {
            this.Tick += onTick;
            this.KeyUp += onKeyUp;
            this.KeyDown += onKeyDown;


            switchLocations.Add(new Vector3(-1578.27f, 5155.2f, 19.79865f)); //Submari
            switchLocations.Add(new Vector3(82.81281f, 6432.408f, 31.31271f)); //
            switchLocations.Add(new Vector3(-237.6f, 5497.5f, 189.6f)); //

            index = 0;
        }

        private void checkDanger()
        {
            if (Game.Player.Character.IsSwimming || Game.Player.Character.IsInCombat |
            {
                //teleport to one of the locations
                Function.Call(Hash.SET_ENTITY_COORDS, Game.Player.Character, switchLoca

                //Move the index to the next location
                index++;
                if (index >= switchLocations.Count) index = 0;
#if (debug)
                UI.Notify("Teleport");
#endif
            }
        }

        private void onTick(object sender, EventArgs e) //this function gets executed
        {
            checkDanger();

            if (botOn)
            {
                //SET PLAYER INVINCIBLE
                Function.Call(Hash.SET_PLAYER_INVINCIBLE, Game.Player, true);

                //SET 1st PERSON VIEW
                Function.Call(Hash.SET_FOLLOW_PED_CAM_VIEW_MODE, 4);

                //TASK SEQUENCE
                TaskSequence mySeq = new TaskSequence();
```

```
                //WALK
                Random rndWalk = new Random();
                int Walk = (int)rndWalk.Next(-10, 10);
                //Game.Player.Character.Task.RunTo(Game.Player.Character.GetOffsetInWorldCoords(
                mySeq.AddTask.GoTo(Game.Player.Character.GetOffsetInWorldCoords(new Vector3(Walk,

                //RUN
                Random rndRun = new Random();
                int Run = (int)rndRun.Next(-10, 10);
                //Game.Player.Character.Task.RunTo(Game.Player.Character.GetOffsetInWorldCoords(
                mySeq.AddTask.RunTo(Game.Player.Character.GetOffsetInWorldCoords(new Vector3(0, R

                //WANDER AROUND
                //Game.Player.Character.Task.WanderAround();
                mySeq.AddTask.WanderAround();

                mySeq.Close();
                Game.Player.Character.Task.PerformSequence(mySeq);

                //LET IT WANDER FOR RANDOM TIME
                Random rndWandering = new Random();
                int Wandering = (int)rndWandering.Next(13, 25);
                Wait(Wandering*1000);
                Game.Player.Character.Task.ClearAllImmediately();
                checkDanger();
                Wait(500);

                if (Game.Player.Character.IsStopped)
                {
                    //PHOTO TAKING STUFF (VIRTUAL KEYS)

#if (debug)
                    UI.ShowSubtitle("take out phone", 1000);
#endif
                    keyOut(0xC8, 1000); //Arrow UP
                    Wait(1000);
#if (debug)
                    UI.ShowSubtitle("move down", 1000);
#endif
                    keyOut(0xD0, 500); //Arrow DOWN
                    Wait(1000);
#if (debug)
                    UI.ShowSubtitle("take left", 1000);
#endif
                    keyOut(0xCB, 500); //Arrow LEFT
```

```
                    Wait(1000);
#if (debug)
                    UI.ShowSubtitle("open app", 1000);
#endif
                    keyOut(0x1C, 500); //ENTER
                    Wait(3000);

                    //RANDOM ZOOM
                    //ZOOM IN
                    Random rndZoomIn = new Random();
                    int ZoomIn = (int)rndZoomIn.Next(4, 14);
#if (debug)
                    UI.ShowSubtitle("Zooooooming In " + ZoomIn + " times", 2000);
#endif
                    for (int i = 0; i < ZoomIn; i++)
                    {
                        mouseWheelOut(130);
                        Wait(20);
                    }
                    Wait(1000);
                    //ZOOM OUT
                    Random rndZoomOut = new Random();
                    int ZoomOut = (int)rndZoomOut.Next(2, 8);
#if (debug)
                    UI.ShowSubtitle("Zooooooming Out " + ZoomOut + " times", 2000);
#endif
                    for (int i = 0; i < ZoomOut; i++)
                    {
                        mouseWheelOut(unchecked((uint)-130));
                        Wait(20);
                    }
                    Wait(3000);

                    //TO DO: SMOOTH RANDOM MOUSE XY
                    /*Random rndCamMove = new Random();
                    int camMove = rndCamMove.Next(5, 10);
                    UI.ShowSubtitle("find the right framing", 1000 + (20 * camMove));
                    for (int i = 0; i < camMove; i++)
                    {
                        Random rndX = new Random();
                        Random rndY = new Random();
                        int newX = rndX.Next(-10, 10);
                        int newY = rndY.Next(-10, 10);
                        //mouseOut(newX*2, newY*2); //mouse xy
                        mouseOut(newX * 10, 0); //mouse xy
```

```
                    Wait(500);
                }
                Wait(2000);*/

                //CHOOSE A RANDOM FILTER
                Random rnd = new Random();
                int filter = rnd.Next(14);
#if (debug)
                UI.ShowSubtitle("choose a random filter", 1000 + (500 * filter));
#endif
                for (int i = 0; i < filter; i++)
                {
                    keyOut(0xD0, 500); //Arrow DOWN
                    Wait(500);
                }
                Wait(1000);

                //TAKE PHOTO
#if (debug)
                UI.ShowSubtitle("take photo", 1000);
#endif
                keyOut(0x1C, 500); //ENTER
                Wait(5000);

                //DELETE PHOTO
#if (debug)
                UI.ShowSubtitle("delete photo", 1000);
#endif
                keyOut(0xD3, 500); //DELETE
                Wait(2000);

                //CLOSE THE CAMERA APP
#if (debug)
                UI.ShowSubtitle("close app", 1000);
#endif
                keyOut(0x0E, 500); //BACK SPACE
                Wait(2000);

                //PUT AWAY THE PHONE
#if (debug)
                UI.ShowSubtitle("put away phone", 1000);
#endif
                keyOut(0x0E, 500); //BACK SPACE

                Wait(2000);
```

```
            }
        }
    }


    //VIRTUAL MOUSE X AND Y MOVEMENT
    public static void mouseOut(int coordX, int coordY)
    {
        //mouse move xy
        Input[] inputs = new Input[]
        {
            new Input
            {
                type = (int) InputType.Mouse,
                u = new InputUnion
                {
                    mi = new MouseInput
                    {
                        dx = coordX,
                        dy = coordY,
                        dwFlags = (uint)(MouseEventF.Move),
                        dwExtraInfo = GetMessageExtraInfo()
                    }
                }
            }
        };
        SendInput((uint)inputs.Length, inputs, Marshal.SizeOf(typeof(Input)));
    }

    //VIRTUAL MOUSE WHEEL SCROLLING
    public static void mouseWheelOut(uint duration)
    {
        //mouse wheel
        Input[] inputs = new Input[]
        {
            new Input
            {
                type = (int) InputType.Mouse,
                u = new InputUnion
                {
                    mi = new MouseInput
                    {
                        dx = 100,
                        dy = 100,
                        //time = 0,
```

```
                        mouseData = duration,
                        dwFlags = (uint)(MouseEventF.Wheel),
                        dwExtraInfo = GetMessageExtraInfo()
                    }
                }
            }
        };
        SendInput((uint)inputs.Length, inputs, Marshal.SizeOf(typeof(Input)));
    }

    //VIRTUAL KEY PRESS AND RELEASE
    public static void keyOut(ushort hexKey, int delay)
    {
        //key pressed
        Input[] inputs = new Input[]
        {
            new Input
            {
                type = (int)InputType.Keyboard,
                u = new InputUnion
                {
                    ki = new KeyboardInput
                    {
                        wVk = 0,
                        wScan = hexKey, //0xC8,
                        dwFlags = (uint)(KeyEventF.KeyDown | KeyEventF.Scancode),
                        dwExtraInfo = GetMessageExtraInfo()
                    }
                }
            }
        };


        //key released
        Input[] inputs2 = new Input[]
        {
            new Input
            {
                type = (int)InputType.Keyboard,
                u = new InputUnion
                {
                    ki = new KeyboardInput
                    {
                        wVk = 0,
                        wScan = hexKey, //0xC8,
```

```
                            dwFlags = (uint)(KeyEventF.KeyUp | KeyEventF.Scancode),
                            dwExtraInfo = GetMessageExtraInfo()
                        }
                    }
                }
            };

            //send key press / delay / release
            SendInput((uint)inputs.Length, inputs, Marshal.SizeOf(typeof(Input)));
            Wait(delay);
            SendInput((uint)inputs2.Length, inputs2, Marshal.SizeOf(typeof(Input)));

        }

        private void onKeyDown(object sender, KeyEventArgs e)
        {

        }
        private void onKeyUp(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.H)
            {
                //SWITCH THE BOT ON/OFF
                botOn = !botOn;
                if(botOn) UI.ShowSubtitle("GTA V Photographer Bot is ON", 3000);
                 else UI.ShowSubtitle("GTA V Photographer Bot is OFF", 3000);

            }
            if (e.KeyCode == Keys.J)
            {
                //TELEPORT MANUALLY
                Function.Call(Hash.SET_ENTITY_COORDS, Game.Player.Character, switchLocati
                index++;
                if (index >= switchLocations.Count) index = 0;
             }
        }
    }
}
```

### 9.0.1 FiveM {–}

//start a server

//join a server

## Content Replication Assignment