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`. The screenshot will be saved to the `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
- In the same panel go to References and click add References… > Browse > browse to Downloads
- Select ScriptHookedVDotNet > `ScriptHookVDotNet2.dll` and `ScriptHookVDotNet3.dll` and add them
- Also add `System.Windows.forms`
- Also add `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, and performing tasks.

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): https://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.