PetraWorld

Marten van Antwerpen  
Antony Löbker  
Nathan van Ofwegen  
Jordy Roelofs  
Furkan Sonmez  
Simon Tiemersma  
Berend Wouda

**30-12-2016**

# 1 Introduction

PetraWorld is a version of ProtoWorld constructed (in part) for the PETRA project. Its main feature is a “dashboard” that provides a user interface to various forms of data visualisation, as well as additional controls for navigating and interacting with the area of interest. It also comes with a number of tools, fixes, and optimizations. Additionally, various scenes have been created using these new features.

The project is stored in the Git (https://github.com/SebastiaanMeijer/ProtoWorld.git) repository, in its own dashboard branch. This branch was created off the develop branch to (further) develop these features in. Some changes have been made to the develop branch as well. For a full overview, one can consult the Git history.

# 2 Modules

PetraWorld adds a number of new modules to the ProtoWorld Editor menu. From here they can be placed into the scene, or their constituents can be placed into the scene. They can also be removed from the scene here.

PetraWorld also involved some changes to existing modules. These are described here as well.

## 2.1 Camera Feed Module

This module can be found in the ProtoWorld Editor menu under Camera Feed Module. Here you can choose to add feed cameras with Add Feed Camera or to remove the module (including all the feed cameras) with Remove Module.

### 2.1.1 Developer

After adding the module (by adding a camera), camera feeds will be displayed at the top of the UI. In the Hierarchy panel a game object named CameraFeedModule will have appeared. Extending it will lead you to the FeedCameras game object, which contains the feed cameras you add. Each feed camera should be placed in the scene as you like it (you can use GameObject🡪Align with view for this). It is best to give it the same rotation as the main camera/overview point combination.

Feed cameras don’t contain any interesting options other than their default ones. It might be necessary to adjust the near and far planes.

Note that feed cameras have a near-empty culling mask. This is to prevent simulation elements from showing up on the camera feeds. Whenever a new layer is added, feed cameras will automatically enable this layer, so they (or their prefab) have to be updated to deal with this.

### 2.1.2 User

During the simulation, the feed cameras will be indicated by icons. Clicking such an icon will take one to the corresponding feed camera.

At the top of the screen are three camera feeds, each displaying a feed camera. All available feed cameras can be scrolled through with the little arrow buttons. Clicking a camera feed will take one to the corresponding feed camera.

## 2.2 KPI Module

This module can be found in the ProtoWorld Editor menu under KPI Module. Here you can choose to add charts with Add Chart or to remove the module (including all the charts) with Remove Module.

### 2.2.1 Developer

KPI modules are partly described in the ProtoWorld Guide. How to set them up can be read there.

A lot of KPIs have been added, for example to the Public Transport Module. Simply drag a module into the feeder to see which KPIs have been added.

KPIs can still be set up visually as before, however, they will organized to the right side of the UI during the simulation. Default colors have been softened. The KPIChart’s ChartController now also has a name option, which renames the chart in the UI.

The Ordering option in the ChartPanel’s ChartPanelController dictates which charts are initially visible and in which order. It also determines how many slots are available.

### 2.2.2 User

During the simulation, the KPIs are represented as graphs that will run alongside the simulation. The chosen KPIs are tracked and displayed.

During the simulation, the KPIs on display can changed in the slot configuration panel. To access this panel, click on the little icon with two gears on them. A list of the currently available slots will be displayed. You can use the dropdowns to select another KPI for that slot. You can use the Add Slot button to add a slot, or the little cross button to remove a slot. Selecting None for a slot will hide it, but not remove it.

The dropdown on the title bar allows you to select a style of visualisation. There are four styles of visualisation.

**Bar** A classic bar chart.

**Line** A line chart that updates over time.

**Pie** A classic pie chart.

**Stacked** Line A cumulative line chart.

The right-most button will fold or unfold the KPI. Similar buttons exist on the top bar, which fold or unfold all KPIs.

The button with the three lines will show or hide the legend. The legend buttons will hide or show specific KPIs within the graph.

## 2.3 Heatmap Module

This module can be found in the ProtoWorld Editor menu under Heatmap Module. Here you can choose to add the module with Add Heatmap Module or to remove the module with Remove Module.

### 2.3.1 Developer

When the module is added, a default heatmap grid is added to the scene. You can find this game object directly under the HeatMapModule game object. Scale this heatmap grid until it covers the area of the scene you wish to have heatmaps for.

Additionally, you can set default values for intensity and radius, in order to tweak the visualisation and to compensate for the scaling. There are also options for management of the heatmap itself, such as its height above the surface or the interval after which it updates.

The count option specifies the maximum amount of data points the heatmap will support during the simulation. Set this to the amount of data points you will expect. Reducing this will improve performance.

Heatmaps are currently fixed in what they can support.

* Pedestrians
  + Locations
* Cars
  + Locations
* Public transport
  + Locations
  + Passengers
  + Delays

Adding additional kinds of data or data points will require adjusting the code.

### 2.3.2 User

A control panel for the heatmaps is available on the left side of the screen. Here you can enable or disable heatmaps globally, as well as tweak their radius to adjust the visualisation.

The type of data point can be selected with the top selection widget. The type of data can be selected with the bottom selection widget.

## 2.4 Historical Data Module

This module can be found in the ProtoWorld Editor menu under Historical Data Module. Here you can choose to add the module with Add Historical Data Module or to remove the module with Remove Module.

### 2.4.1 Developer

Historical data is logged every preset amount of seconds into an XML document. Upon the user’s request, this document can be saved to disk using the current timestamp as file name. They will be saved to the log directory. The user can then specify to load this data at a future time.

Historical data is currently retrieved and restored through a system which sports the Loggable interface and a LoggableManager as a registry. Components can implement the Loggable interface to load and save data using the Historical Data Module. See these classes for more information.

Currently the following modules are supported.

* FLASH Pedestrian Module
* Public Transport Module
  + Busses are not supported.

The Historical Data Module has only one option, the (repeated) interval after which to log. Logging can be a costly operation, but it is normally only performed once in a while.

### 2.4.2 User

During the simulation, you can click the Save button to save the current (most recently stored) game state. As long as the scene remains the same (or similar enough), this log can be loaded to restore (parts of) the game state to the stored state with the Load button.

The Load button opens a file browser, that lists all available log files (from the log directory) on the left-hand side. Selecting one will list all available times the state was stored, and you can select one of these to load. Click the Load button to load the state. You can close this screen without loading anything by clicking the little cross button.

Note that the simulation will pause while you are in the file browser.

## 2.5 Issues Module

This module can be found in the ProtoWorld Editor menu under Issues Module. Here you can choose to add issues with Add Issue or to remove the module (including all the issues) with Remove Module.

### 2.5.1 Developer

Issues appear under the Issues game object inside the module. They have no options but their child, the sign, has a few.

**Issue Type Text** The type of issue being reported. For example, traffic jam or accident.

**Details Text** A short description of the issue. This will displayed along with the type in a popup menu when the issue is clicked.

**Scale** The SpriteController controls the look of the sign. You can use it to make it appear smaller or larger.

**Hide Distance** Similarly, this can be used to hide the issue when the camera gets too close.

The Issues game object itself contains a RandomIssueSpawner. This is mostly for demonstration purposes. The interval at which it spawns issues can be set as an option.

### 2.5.2 User

During the simulation, exceptional situations can occur which are indicated with a yellow warning sign.

Clicking the sign will display a dialog that tells you the type of the issue and what exactly it entails. From here you have several choices.

1. Ignore the issue by clicking outside the dialog. The issue will not disappear.
2. Go to the issue by clicking the Go To Issue button. The camera will focus on the issue. The issue will not disappear.
3. Solve the issue by clicking the Solve Issue button. The issue will disappear.

## 2.6 Micro Macro Visualisation Module

This module can be found in the ProtoWorld Editor menu under Micro Macro Visualisation Module. Here you can choose to add the module with Add Micro Macro Visualisation Module or to remove the module with Remove Module.

### 2.6.1 Developer

The Micro Macro Visualisation module works by changing the main camera’s culling mask. In the MicroMacroVisualisation game object you can configure this behaviour.

In the Layer Levels option, enter how many levels you wish to configure. Per configuration, you can indicate the layer or layers to hide once the camera reaches that level (index). If you check the Inverted checkbox, the layer will be shown instead (and hidden until the camera reaches the level). Note that this assumes you are moving the camera down (“zooming in”). If you are moving the camera in the opposite direction, it will work the other way around.

Note that the level index you enter is inclusive, the layer will be shown once that level is reached. Also note that the level index is an index into the ScrollbarMMV’s Stages Heights (and Display Text) arrays.

You can also set this up for individual game objects (and their children) using the Object Levels option. This is useful for hiding or showing UI elements.

The ScrollbarMMV’s Stages Heights option determines at which camera heights the levels switch. The Display Text option provides their names. You can configure any number of levels.

### 2.6.2 User

At the right side of the screen you will find a scrollbar that you can use to move the camera closer to and further away from the scene. This scrollbar also indicates the available levels and at which heights they reside. Once you pass into a level, the scrollbar will briefly flash its name.

Once you pass into a level, things may disappear or appear depending on the scene and its configuration. This is used to provide the right visualisations for the current level of detail.

The checkbox on top of the scrollbar disables the micro/macro visualization.

## 2.7 Overlay Module

This module can be found in the ProtoWorld Editor menu under Overlay Module. Here you can choose to add the module with Add Overlay Module or to remove the module with Remove Module.

### 2.7.1 Developer

The Overlay Module can be configured in the OverlayUI game object. It simply requires a list of game objects you wish to enable or disable when the overlay button is clicked.

Additionally, you can specify the text for the button in either state.

### 2.7.2 User

At the bottom of the screen you may find a large button. Clicking it will enable or disable certain parts of the user interface. Clicking it again will return the user interface to what it was before.

## 2.8 Flow Visualisation Module

The Flow Visualisation Module is an experimental feature not available from the ProtoWorld Editor menu. It can be found in Assets/Assets ProtoWorld/Flow Visualisation.

### 2.8.1 Developer

This module is operational but still needs work. Its visualisation is limited and probably not what was intended. Additionally, large amounts of data produce very difficult to discern line patterns.

It is also hardcoded for two specific use cases, and can only use data provided from within Unity itself. This means builds are limited to what was configured.

Its options include a list of text assets, which should be CSV files in either Haifa or Venice formats, for which the dropdown should be set correctly. Venice only takes two files, the points of interest and the travel times. Haifa can take any number of files, and assumes they are in chronological order.

The Sample option specifies how many points of interest to use in the Venice case (it has no effect on the Haifa case). This is to reduce the huge amount of data to a visible demonstration.

The Line Width and Lane Width options control the display of the lines. Higher settings are better for larger spaces.

### 2.8.2 User

The slider at the bottom of the screen will scroll through the available time steps. The colors of the lines will change according to their value (generally a count or another measure of how busy that route is).

## 2.9 Scenario Module

The ScenarioModule has had few changes. The surface mesh has been scaled up to provide a screen-filling background (sometimes scenes even scale it up further), so that the color scheme is the same everywhere.

## 2.10 Navigation Module

The NavigationModule has undergone a few changes. It is the central interactive module of every scene, and as such the best place to put generic scene-related functionality. Part of it is part of the UI as well, which was overhauled as part of the dashboard addition. Finally, scenes sometimes required camera changes that made more scene as a generic change.

### 2.10.1 Buttons

The button to reset the camera has been resized and placed in a panel with a few other buttons.

* The button with the world icon returns the user to the menu.
* The button with the cross icon exits the application to the desktop.
* The button with the camera resets the camera.

### 2.10.2 Main Camera

The main camera underwent a few changes.

* Due to the addition of the layers and the Micro Macro Visualisation Module, the camera has a richer culling mask.
* The camera has a hidden movement feature that can be accessed by dragging with the middle mouse button. It was added for development purposes.
* The compass has been disabled.

### 2.10.3 Time controller

The TimeController underwent a visual overhaul to reduce the space it takes up. This way there was more space for other elements in the dashboard.

### 2.10.4 Transition controller

The TransitionController is a new feature that fades a scene in from a solid color (the default is white). This provides smooth transitions between scenes (including the menu) as well between a scene and the Unity logo (if built on a free version).

By default it is disabled. In the TransitionUI game object, check Use Transition to enable transitions.

If transitions have been enabled it can also fade a scene out to white, if requested by another script (currently the menu, and the Return to Menu and Exit Application buttons use this functionality).

## 2.11 Pedestrian Module

The FLASH Pedestrian Module was amended quite a bit under the hood to support the Historical Data Module. It also had some performance optimizations implemented, and its UI has been compacted to save space.

## 2.12 Public Transport Module

The Public Transit Module has had some changes to its code to support the heatmaps and the historical data.

Additionally, the station dialog has been moved to a central location together with a number of other dialogs.

Finally, the text is now automatically rotated and resized by a TextController so that it always appears the same on the screen.

## 2.13 Traffic Integration Module

The Traffic Integration Module has had some changes to its code to support the heatmaps.

Also, the default angle of cars has been reset to SUMO angles.

## 2.14 Drag and Drop Module

The Drag and Drop Module was moved to a different location as part of the dashboard changes.

# 3 Scenes

Various scenes have been released as part of the PETRA project.

## 3.1 Rome

The Rome case is a large scene which showcases almost every module created as part of the PETRA project. It contains all modules except for the experimental Flow Visualisation Module.

The Rome scene is best left explored by oneself. For details on its functionalities one can see the sections on each module.

## 3.2 Haifa

The Haifa case is a smaller scene that focuses on the city in Israel. It contains a visualization of their Bluetooth sensor network, which counts passing cars to build an overview of road use.

This scene uses only the ProtoWorld Essentials, as well as a minimalistic Micro Macro Visualisation Module and an experimental Flow Visualisation Module.

The Flow Visualisation Module displays a (prerecorded, averaged) sample of Bluetooth sensor data which can be explored through time by using the slider at the bottom of the screen.

For more information on the flow visualisation, see the section on the Flow Visualisation Module.

## 3.3 Venice

The Venice case is the smalleset scene. It focuses solely on the island lagoon in Italy. It contains a visualization of an external pedestrian simulation.

This scene uses only the ProtoWorld Essentials, as well as a minimalistic Micro Macro Visualisation Module and an experimental Flow Visualisation Module.

The Flow Visualisation Module displays a (pregenerated) simulation which can be explored through time by using the slider at the bottom of the screen. It only displays a (random) sample of all available points of interest each time, to reduce the clutter of the many lines available.

This visualization is not very effective. A different kind of visualization is needed (with more interaction). See also the recommendations.

For more information on the flow visualisation, see the section on the Flow Visualisation Module.

## 3.4 Menu

The menu scene is a small scene without module functionality or even a controllable camera. It contains a simple, fully configurable menu that can be used to release multiple scenes at once.

### 3.4.1 Script configuration

The menu prefab can actually be placed in any scene. This way one can create a menu from the ground up, if one wishes to do so.

In order to configure the menu, extend the menu game object in the scene until you find the MenuUI game object. Attached to this object is the Menu script, which contains all the configuration options for the menu.

**Title**  The name of the product. This appears at the top of the menu.

**Version** The version of the product. This appears at the top of the menu, next to the name. This could be left empty.

**Menu Items** The various entries in the menu. These are clickable images with a title that will load and start a scene.

**Title** The title of the entry. This is displayed below the image.

**Image** A texture to serve as the displayed image. This can be left empty, a default image will then appear.

**Path** The path to the scene in the Unity project directory. For example Cases ProtoWorld/Rome Case/Rome Case v2.0.

**Transition Controller** Set this to the scene’s TransitionController if you wish to use transitions.

### 3.4.2 Scene configuration

If you build a menu from the ground up, we suggest disabling any UI canvasses as well as the camera control scripts and aiming the main camera at a place of interest.

### 3.4.3 Unity configuration

Note that in order to release multiple scenes, each scene (and the menu scene) has to be checked in the build settings dialog. The menu should be the first checked scene in the list.

## 3.5 Default

A default test scene has been added (in Assets/Miscelanea/Tests/DefaultTest) that sports the new color scheme. It only contains the two essential modules, the generated meshes, and a navigation mesh. It is/was intended for creating new test scenes from.

# 4 Utilities

Through development various utilities and tools have been developed. They can be found in the ProtoWorld Editor menu.

The existing tools have also been edited, although to a minimal extent. The changes only regard map generation for PetraWorld maps (assignment of layers and so on), and some bugfixes here and there.

## 4.1 Post-generation utilities

For certain modules, PetraWorld requires additional processing after a map has been generated using the regular tool chain.

### 4.1.1 Remove Buildings with Vertices with Non-finite Values

This tool can be found in the ProtoWorld Editor menu under ProtoWorld Essentials/Map Tools/Utilities/Remove Buildings with Vertices with Non-finite Values. It removes buildings that have a mesh with one or more vertices in it that contains a non-finite value (infinity or not-a-number). Meshes with such a vertex are sometimes generated from OpenStreetMap data, and cause Unity to produce a lot of errors.

### 4.1.2 Remove Roads with Vertices with Non-finite Values

This tool can be found in the ProtoWorld Editor menu under ProtoWorld Essentials/Map Tools/Utilities/Remove Roads with Vertices with Non-finite Values. It removes roads that have a mesh with one or more vertices in it that contains a non-finite value (infinity or not-a-number). Meshes with such a vertex are sometimes generated from OpenStreetMap data, and cause Unity to produce a lot of errors.

### 4.1.3 Remove Empty Buildings

This tool can be found in the ProtoWorld Editor menu under ProtoWorld Essentials/Map Tools/Utilities/Remove Empty Buildings.

Once buildings have been combined with other buildings for performance reasons, their mesh colliders can be removed using the existing Remove Building Colliders tool. This tool will then remove the remaining building game objects that no longer have a mesh or a collider. This removes a lot of game objects that are no longer needed, and improves performance even further (as well as file size).

### 4.1.4 Remove Empty Roads

This tool can be found in the ProtoWorld Editor menu under ProtoWorld Essentials/Map Tools/Utilities/Remove Empty Roads.

Once a navigation mesh has been built and roads have been combined with other buildings for performance reasons, their mesh colliders can be removed using the existing Remove Road Colliders tool. This tool will then remove the remaining road game objects that no longer have a mesh or a collider. This removes a lot of game objects that are no longer needed, and improves performance even further (as well as file size).

## 4.2 Upgrades

In order to upgrade existing ProtoWorld maps to be used with PetraWorld functionality (or older PetraWorld maps), a number of tools are available.

### 4.2.1 Assign Missing Layers to Buildings

This tool can be found in the ProtoWorld Editor menu under ProtoWorld Essentials/Map Tools/Upgrades/Assign Missing Layers to Buildings. Running it assigns the Building layer to building game objects.

The Building layer is used to arbitrarily hide all buildings at once by adjusting the main camera's culling mask. This is used as part of the Micro Macro Visualisation module.

### 4.2.2 Assign Missing Layers to Roads

This tool can be found in the ProtoWorld Editor menu under ProtoWorld Essentials/Map Tools/Upgrades/Assign Missing Layers to Roads. Running it assigns the TrafficRoads, footways, or steps layer to any road game objects missing them.

Road game objects normally have these layers already assigned to them as part of the map generation process, so this tool is only necessary in exceptional situations. It has been provided for symmetry purposes.

The layers are primarily used to build the navigation mesh but they can also be used to arbitrarily hide all roads at once by adjusting the main camera's culling mask. As such this can be used as part of the Micro Macro Visualisation module.

### 4.2.3 Assign Missing Layers to Grouped Buildings

This tool can be found in the ProtoWorld Editor menu under ProtoWorld Essentials/Map Tools/Upgrades/Assign Missing Layers to Grouped Buildings.

Once all buildings have been combined into cells based on their location and material, they will be lacking their original layers. This tool restores them based on their material.

Unfortunately the Group Close Buildings tool does not perform this function for you at the moment. We originally upgraded it, but it doesn't work right due to timing issues when building the combined mesh (it is delayed). This is the “quick” solution, however the proper solution could perhaps still be implemented. The original code is available as a TODO comment in the PrepareForCombining.cs file.

### 4.2.4 Assign Missing Layers to Grouped Roads

This tool can be found in the ProtoWorld Editor menu under ProtoWorld Essentials/Map Tools/Upgrades/Assign Missing Layers to Grouped Roads.

Once all roads have been combined into cells based on their location and material, they will be lacking their original layers. This tool restores them based on their material.

Unfortunately the Group Close Roads tool does not perform this function for you at the moment. We originally upgraded it, but it doesn't work right due to timing issues when building the combined mesh (it is delayed). This is the “quick” solution, however the proper solution could perhaps still be implemented. The original code is available as a TODO comment in the PrepareForCombining.cs file.

Note that different types of road have different types of materials. Because the tool restores layers based on materials, the layers are assigned correctly across the board.

### 4.2.5 Add Missing Camera Sprite Controllers

This tool can be found in the ProtoWorld Editor menu under ProtoWorld Essentials/Map Tools/Upgrades/Add Missing Camera Sprite Controllers. It adds SpriteControllers to feed cameras (added by the Camera Feed Module) so their icon displays correctly.

It is only of interest for outdated PetraWorld scenes. New cameras are generated correctly, because the prefab has been updated.

### 4.2.6 Add Missing Issue Sprite Controllers

This tool can be found in the ProtoWorld Editor menu under ProtoWorld Essentials/Map Tools/Upgrades/Add Missing Issue Sprite Controllers. It adds SpriteControllers to issues (added by the Issues Module) so their icon displays correctly.

It is only of interest for outdated PetraWorld scenes. New issues are generated correctly, because the prefab has been updated.

### 4.2.7 Add Missing Station Text Controllers

This tool can be found in the ProtoWorld Editor menu under ProtoWorld Essentials/Map Tools/Upgrades/Add Missing Station Text Controllers. It adds TextControllers to stations (added by the Public Transport Module) so their text displays correctly.

It can be used for original ProtoWorld scenes or outdated PetraWorld scenes. New stations are generated correctly, because the prefab has been updated.

## 4.3 Materials

These tools assign the PetraWorld color scheme to the scene.

### 4.3.1 Replace materials

This tool can be found in the ProtoWorld Editor menu under ProtoWorld Essentials/Map Tools/Materials/Replace materials. It is used to replace the materials on meshes of game objects in the scene with another material.

Selecting the tool pops up a dialog window requesting a tag and a material. The tag can be selected from the dropdown, and the material can be selected from the finder or by dragging it from the assets library.

The Replace Materials button will replace the materials of the meshes of all the game objects with the selected tag to the selected material. It also attaches an OriginalMaterials script to the game object, which is used to store the original materials of its mesh. The Restore Materials button will replace the materials of the meshes of all the game objects with the selected tag with their original materials as specified by their OriginalMaterials scripts, and remove the OriginalMaterials scripts.

There are a number of default replacement materials used for the scenes in PetraWorld. These can be found in Assets/Miscelanea/Resources/Replacement Materials.

**Area** For walkable areas.

**Building** For buildings.

**Road** For traffic roads, footways, and steps.

**Surface** For the surface of the earth.

**Water** For water areas and imported water meshes.

# 5 Optimizations

Various performance optimizations have been added to ProtoWorld as part of the PetraWorld development. These were mostly to improve performance on our development machines, after identification during profiling.

## 5.1 Pedestrians

The FLASH Pedestrian Module had a few bottlenecks that have been alleviated. Do note though that, from our experience, performance still scales inversely to the number of pedestrians in the scene.

### 5.1.1 Itinerary search

When a pedestrian spawns, it decides on a destination, and it performs a search through all possible itineraries in order to find the most optimal itinerary to its destination according to its travel preferences. During this search it performed a sort of all the itineraries in order to find the best one. Because many pedestrians can spawn at the same time, this sort was performed a large number of times in the same frame, leading to stutter. Replacing the sort by a minimum search seemingly removed this performance hit.

### 5.1.2 GameObject.Find

When a pedestrians spawns, it uses the GameObject.Find method to search for various objects in the scene required for proper operation of the pedestrian scripts. This method (and its family of methods) is notorious for being a heavy operation. Normally, executing these in Awake() or Start() is fine, as they run only at the start of the scene. Pedestrians can spawn at any time however, and spawn in the same frame in potentially large numbers, so these method calls end up being executed a lot during a single frame. They all find the same game objects, so these game objects are now found and cached by the spawner at the start of the scene, and are provided to the pedestrians when they spawn instead.

## 5.2 Materials

Various instances of materials being created during spawn events have been replaced with a material being created only once, and then cached for further spawn events. Creating a material is a heavy operation, as communication with the videocard is involved.

## 5.3 KPIs

Although KPIs were added as part of the dashboard, it might be of interest to mention the approaches taken to minimize performance impacts.

Essentially, KPIs can be gathered in a centralized manner, or a decentralized manner. A decentralized manner is often more performant.

Collecting data for a KPI often requires including all game objects of some kind, for example all pedestrians. This creates a dependency at least proportional to n. Since Unity already visits all of these game objects for their updates, performing the data collecting as part of their update reduces the load somewhat. The data is then pushed instead of pulled, which would entail looping over all the pedestrians in the update of another script. The data is generally pushed to a central place, which the KPI module is fed from once per frame by another script.

One example is the amount of pedestrians using a bike. Instead of looping over all pedestrians to inspect if they were riding a bike in a KPI script, their state machine now reports when a pedestrians gets on or off a bike. This appears to removes the associated bottleneck.

# 6 Changes

Unfortunately the ProtoWorld project as delivered to us did not work out of the box. Various changes had to be made, mostly to the project setups, to make the project operational. These changes have, at the time, mostly been done in the develop branch. Some changes have been performed after the branching into the dashboard branch, most notably a reverse commit of a merge of the develop branch into the dashboard branch, as it rendered the project nonoperational.

## 6.1 .gitignore

The project's various .gitignore files have been altered to allow for some files or file types, and to disallow some others. These changes have almost all been made on the dashboard branch.

### 6.1.1 .meta

Most notably, Unity's .meta files have been allowed (and subsequently committed) onto the repository. Otherwise Unity will regenerate these files by its own accord. This addresses issues where relations between assets were lost.

## 6.2 LFS

GitHub's Large File Storage initially caused us a lot of trouble. What exactly originally caused this is not entirely known (it might have been use of outdated LFS plugins at some point), but it caused issues such as the inability to switch branches, missing files, and the inability to pull from the central repository.

Changes have been made to the LFS setup that are mostly reflected in the .gitattributes file. They move certain file types to normal Git storage, and certain file types into Git LFS storage. The recommitting of various files after these changes (sometimes requiring a remove and re-add) generally provides the actual fix.

Although the issues have been resolved, with GitHub it is still possible that LFS sometimes will not pull files correctly. This is because the smudge filter fails silently upon an authorization step. GitHub does not require authentication for LFS enabled on public repositories, but LFS can sometimes get confused about this for some reason, and when it does, it adds a line to the project's .git/config file. This line causes the smudge filter to fail. Removing this line will restore pull functionality. The line to remove is under the LFS configuration section for the repository ([lfs "https://github.com/SebastiaanMeijer/ProtoWorld.git/info/lfs"]), and specifies the access mode (access=basic).

These changes have all been made on the develop branch.

## 6.3 Scripts and tools

A number of the various C# projects that come with the ProtoWorld Unity project (in ScriptsAndTools and WCFService) turned out not to be operational. This was mostly caused by absolute paths and references to resources not in the project repository. Before we could build scenes, we had to address these issues, because otherwise we could not build the software required. This required changing the project configuration files. Some files were moved as well.

These changes were executed mostly on the develop branch, but a later commit reversed some of these changes. The dashboard branch later had the develop branch merged into it, and as a result this reversing commit as well. Since this made the tools nonoperational again for us, the dashboard branch now also contains a reverse commit of this commit. Dealing with this is important if a merge ever occurs, see the section on merging for more information on this.

## 6.4 Unity project

Various aspects of the Unity project configuration has been altered to fit our needs.

### 6.4.1 Player settings

The API back end has been set to Mono2x. This is necessary for the project to build and presumably the standard, but the player settings were originally not stored in the repository.

### 6.4.2 Graphics settings

The default graphics settings have been removed and replaced with only one option, a full HD resolution with high graphics settings and anti-aliasing enabled. PetraWorld targets only a few specific machines which support this. As it stands, the user interface does not scale or resize dynamically, and has been designed for a full HD resolution.

### 6.4.3 Build settings

The configuration dialog that normally pops up before playing a build has been disabled. PetraWorld targets only one graphics option, and the dialog popping up makes demos less fancy. It is, however, trivial to turn it back on.

## 6.5 Git

A number of branches have been created in the Git repository aside from the dashboard branch.

Most features have been implemented in separate branches, often grouped under branch directories. Essentially, the dashboard branch has been treated as our develop branch. Some of these branches remain, but upon the time of delivery all they should all have been merged into the dashboard branch, if their changes were desired for the final product. If their changes have not been merged they can be discarded, unless otherwise specified during the hand-over.

A branch of note is the release branch, which performs the function of a master branch if you think of the dashboard branch as a develop branch. It contains all commits up until the latest major release. At the time of the hand-off all changes should have been merged into this branch.

Various tags exist as well to indicate the major releases. These are named “PetraWorld” followed by a release number (the version).

# 7 Conclusion

This document has aimed to give you an overview of the changes and additions made to ProtoWorld to create the dashboard variation. It is not an exhaustive document, but covers all the important changes. Any remaining changes can be looked for in the Git history of the dashboard or release branch of the repository.

## 7.1 Recommendations

Currently, there are few recommendations to be made. PetraWorld is rather feature-complete and we have aimed for a full-bodied final release. However, there is one topic that benefits from discussion.

### 7.1.1 Merging

At the time of writing, the dashboard branch is considered separate from the develop branch, and whether or not the dashboard branch will ever be merged into the develop branch is unclear. It has to be decided first if this is the direction ProtoWorld should be taken in.

Regardless, a case can still be made for a merge of the dashboard branch into the develop branch. As described in this document, the changes made for PetraWorld were either in a modular fashion, or under the hood without changing existing behaviour, or to the various project configurations.

The first two allow for ProtoWorld to continue being used as it is. Modules do not have to be used, or can be used individually, and can generally be configured to fit the current need. Changes that only work under the hood, either to assist modules if they are active, or as a performance optimization or a bug fix, are not visible to the end user and as such provide the same experience. Therefore, merging these kinds of changes into the main ProtoWorld development only adds optional functionality. Additionally, it is fairly easy to disable the optional functionality, as the modules and utilities can be removed from the ProtoWorld Editor menu quite easily.

The last one is, in our view, also beneficial to the develop branch. These are changes we had to make to get a freshly cloned repository to work right off the bat. We feel that this improves ProtoWorld in an acceptable manner. However, it is fairly easy not to merge these kinds of changes into the main ProtoWorld development, as they are to files outside of the Unity project.

The merge itself can be quite daunting. The dashboard branch has branched off the develop branch several months ago, and although not many changes have been committed to the develop branch, they are still changes to files also changed in the dashboard branch. This will likely result in merge conflicts, which will then have to be resolved. Normally, this is handled piecemeal for the branch created from the develop branch, in order to minimize merge efforts in the future. However, the unclear status of a future merge, combined with a merge issue that arose early on, meant that the dashboard branch has not been kept up-to-date with the develop branch.

The merge issue just mentioned made it difficult for us to continue merging changes of the develop branch into the dashboard branch. Changes that were made to the develop branch before the dashboard branch was created were reversed in a later commit. This later commit was merged into the dashboard branch as part of our piecemeal merging process, rendering it nonoperational. We had to reverse these changes in a commit to the dashboard branch to restore everything to working order.

Merging the dashboard branch back into the develop branch would also merge this reverse commit into the develop branch. This essentially undoes a commit on the develop branch. On the one hand, this commit appeared to break the project configuration for us. On the other hand, it may contain more changes not related to the project configuration. Perhaps the best solution here is to create a reverse commit of the reverse commit made on the dashboard branch, and then figure out whether to reapply the changes to the project configuration afterwards. After all, these are part of the third category of changes mentioned above.

If performing a merge, this is an idea of how we would suggest doing it. This method can also be used to perform a test merge, to see if it is feasible and to investigate whether it provides a desired version of ProtoWorld.

1. Create a develop-with-forced-text branch from the tip of the develop branch.
2. Check out the develop-with-forced-text branch and open Unity. In the project settings, find the file format option. Set it to Force text. Wait while Unity changes all the files. Then commit the changes to the develop-with-forced-text branch.
3. Create a dashboard-with-forced-text branch from the tip of the dashboard branch.
4. Check out the dashboard-with-forced-text branch and open Unity. In the project settings, find the file format option. Set it to Force text. Wait while Unity changes all the files. Then commit the changes to the dashboard-with-forced-text branch.
5. Create a merged branch from the tip of the dashboard-with-forced-text branch.
6. In the merged branch, find commit 44bd2b8eb0b3b2ec4780b9912e7602f8291ec9d9 and perform a reverse commit of this commit.
7. Merge the develop-with-forced-text branch into the dashboard-with-forced-text branch.
8. Inspect the results. If merging did not occur correctly due to the various prefabs not being merged as text, reset all changes to the working copy and the index, open the .gitattributes file and change all instances of -unityyamlmerge to -text. Save it, and try the merge again.
9. Fix any merge conflicts. Whether any should arise we cannot predict, nor do we know how exactly the prefabs should be merged. This will require some figuring out.
10. Make a decision about whether to reapply the changes made in commit 44bd2b8eb0b3b2ec4780b9912e7602f8291ec9d9.
11. Make a decision about whether to apply the changes outside the Unity project folder. There are various ways to not include these changes, some of which could be done during the merge instead. This will require some figuring out.
12. Commit the resulting merge to the merged branch.
13. Open Unity. In the project settings, find the file format option. Set it to Mixed. Wait while Unity changes all the files. Then commit the changes to the merged branch. Alternatively, you can keep it set to Force text. This will create larger files, but is easier for merging and bugfixing. The increased size is offset by Git's (LFS) capability to compress files during transfer.
14. If ProtoWorld looks and behaves as desired, one can merge the merged branch into the develop branch.
15. Push the develop branch to the remote repository.
16. Delete all created branches and optionally even the dashboard, release, and related feature branches. If you delete any of the latter, you will have to push your deletions to the remote repository.