Platform Perils

User Guide

Steven Palmer
⟨palmes4⟩
Chao Ye
⟨yec6⟩

April 26, 2016

Contents

List of Tables

List of Figures

Revision History

Date	Version	Notes
February 28, 2016	1.0	Created document
February 29, 2016	1.1	Finished draft
February 29, 2016	1.2	Rev 0 final
April 25, 2016	1.3	Rev 1 final

1 Legal and Copyright Information

The Platform Perils software and all related materials are owned by McMaster University (excluding external libraries and textures). By using the software you accept all liability for any damages caused by the software.

2 Introduction

The purpose of this guide is to provide installation instructions as well as an overview of the game. The guide is structured as follows:

- The first part of the guide deals with system requirements and offers step-by-step installation instructions (see §3).
- The second part of the guide serves as an instruction manual for the game (see §4, §5)
- The final part of the guide provides additional help in the form of FAQ and troubleshooting sections (see §6, §7)

2.1 Definitions

The definitions used throughout this guide are listed in Table 1.

Table 1: List of Terms

Term	Definition
`installer`	Refers to the `gamebuilder.tar.gz` archive that contains an automated installer and the files required to build the game. When extracted, a folder structure with a root directory called `gamebuilder/` is created.
`game directory`	The game directory can be found at `gamebuilder/game/` (see `installer`)
`source directory`	The source directory can be found at `gamebuilder/source/` (see `installer`)
`build directory`	The build directory can be found at `gamebuilder/source/build/` (see `installer`)

3 Getting Started

3.1 System Requirements

3.1.1 Hardware Requirements

Running the game requires a system with a GPU that supports OpenGL v3.3 or higher.[1] Hardware requirements related to performance have not been assessed.

3.1.2 Software Requirements

Platform Perils is compatible with the following operating systems:

- `Windows 7`
- `Mac OS X 10.6 and higher`
- `Linux (confirmed working on the Arch Linux distribution)`

[1] Consult the specifications of your GPU to determine the highest supported version of OpenGL.

The following software will be required to install the game:

- `CMake`
- `gcc`
- `make`

Please refer to the installation instructions in the following section for further details.

3.2 Installation Instructions

A cross-platform compatible installer is included with the game. To use the installer you will need to download and install the latest release of CMake. Make sure that **CMake** can be run from the terminal/command prompt (you may need to edit your PATH variable).

3.2.1 Installing on Windows

1. Download and install mingw-w64, which will be used to compile the game and the required libraries. Install with default settings (threads should be set to POSIX) and ensure that **gcc.exe**, **g++.exe**, and **mingw32-make.exe** can be run from the command prompt (you may need to edit your PATH variable).[2]

2. Open a command prompt and navigate to the pre-made `build` directory (`gamebuilder/source/build/`).

3. Run the command

   ```
   cmake -G "MinGW Makefiles" ..
   ```

   to invoke cmake. A makefile (as well as many other files) will be generated in the `build` directory.

[2] **mingw-w64** is a fork of **mingw** and much more up to date. Regular **mingw** installations do not include libraries that are required to build the game and its required libraries. If you have a previous installation of **mingw** you can either replace it or keep both versions.

4. Run the command

   ```
   mingw32-make
   ```

   to begin building the libraries and the game. This will build all of the necessary libraries as well as the game. The build process should take 1 to 2 minutes.

5. Run the command

   ```
   mingw32-make install
   ```

   to install the game to the `game` directory. All of the files required to run the game will be moved to this directory.

3.2.2 Installing on Mac OS X and Linux

1. Open a terminal and navigate to the pre-made `build` directory (`gamebuilder/source/build/`).

2. Run the command

   ```
   cmake -G "Unix Makefiles" ..
   ```

   to invoke cmake. A makefile (as well as many other files) will be generated in the `build` directory.

3. Run the command

   ```
   make
   ```

   to begin building the libraries and the game. This will build all of the necessary libraries as well as the game. The build process should take 1 to 2 minutes.

4. Run the command

```
make install
```

   to install the game to the `game` directory. All of the files required to run the game will be moved to this directory.

3.2.3 Other Operating Systems

The game must be installed manually on other operating systems. To perform a manual installation you will need to download and compile the following libraries:

1. Chipmunk2D
2. GLFW
3. GLEW
4. OpenAL Soft
5. FreeALUT

The game can then be built using the source files and headers located in the `src` and `include` folders in the `source` directory. Be sure to merge all of the include folders that come with the libraries with the `include` folder found in the `source` directory folder. All of the aforementioned libraries must be linked when compiling the game (as well as any operating system dependent libraries you may require).

3.3 Running the Game

Once the game is installed, it can be run via the executable found in the `game` directory. When running the game for the first time, use of the terminal/command prompt is recommended in case any error messages arise. If an error is encountered, refer to §7 for troubleshooting.

4 Game Basics

Platform Perils focuses on a nameless hero who finds himself lost in a world full of dangerous hazards. It is up to you to help the hero navigate safely through a series of perilous stages so that he can return home.

4.1 Main Menu

The game begins on a main menu screen from which stages can be selected to be played. Select a stage by clicking on it to begin.

4.2 Stages

Each stage in the game begins with the hero situated at a designated start position. If the hero is killed by a hazard in the stage, he will restart at the start position. The goal of each stage is to reach the arch (see Figure 1) on the checkered platform while avoiding the deadly hazards. The game stages become progressively more difficult.

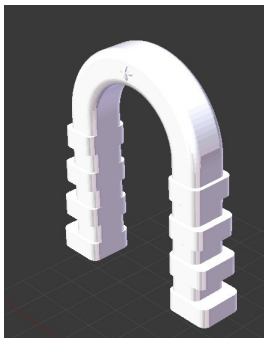

Figure 1: End of stage arch.

4.2.1 Hazards

The hazards that will be encountered in the game stages are summarized in Table 2. The visual representation of each hazard is given in Figure 2.

Table 2: Hazards

Hazard	**Description**
Boulder	The boulder is a dangerous hazard that will crush any person in its path. When the boulder has stopped rolling is no longer hazardous and may even be used as a platform.
Spikes	Spikes are a fatal hazard that are frequently found on the tops and bottoms of platforms. Never touch them!
Spears	Spears are The

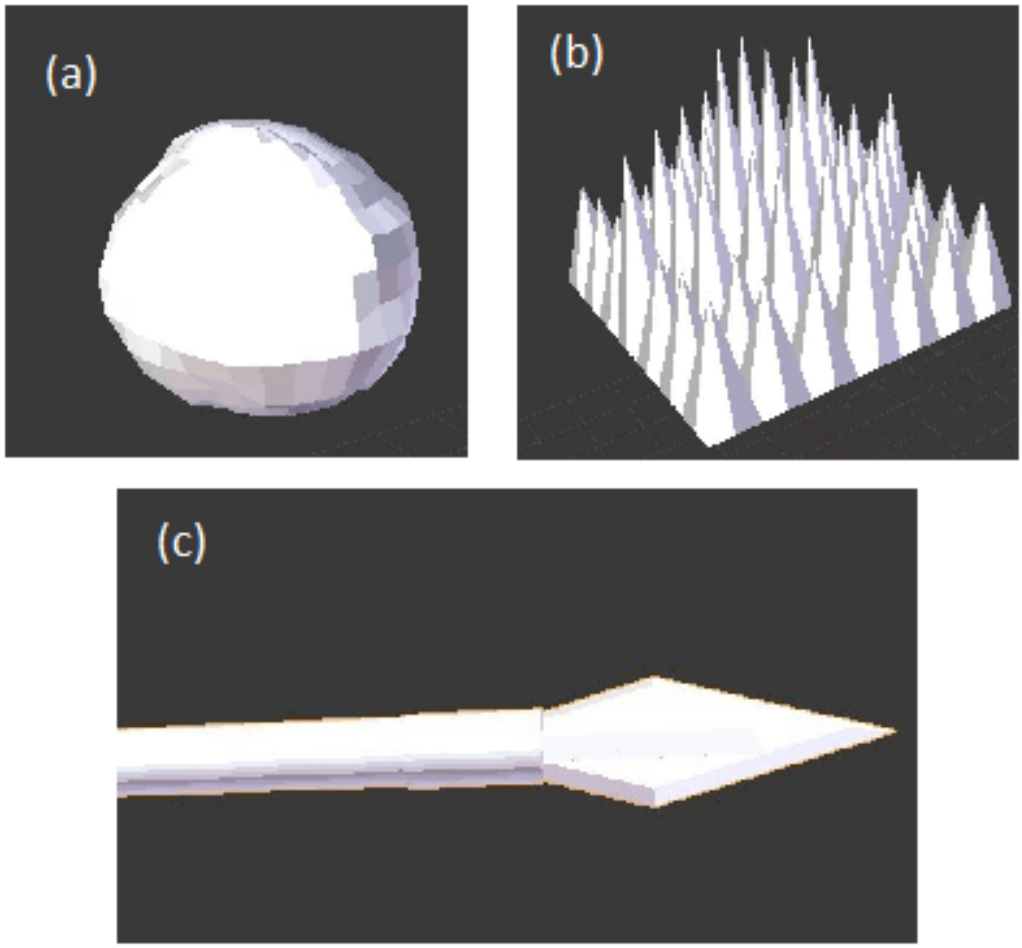

Figure 2: Hazards found in the game. (a) Boulder. (b) Spikes. (c) Spear.

4.3 Game Controls

The default game controls are given in Table 3.

Table 3: Game Controls

Key	Function
A	Move left
D	Move right
SPACE	Jump
W	Zoom in
S	Zoom out

5 Level Scripts

Once you have finished the three stages that come with game, the fun continues! Stages are defined in level scripts found in `/data/stage/` (starting from the `game` directory), which can be edited to create custom stages. The remainder of this section provides information about the level script format.

5.1 Basics

The shorthand used to explain script syntax is given in Table 4. The fields encountered in the scripts are given in Table 5.

Table 4: Shorthand used to explain script syntax

Shorthand	Meaning	Example
{opt1—opt2—opt3}	Choice	opt1 **or** opt2 **or** opt3
opt1[, opt2]	Optional	opt1 **or** opt1, opt2
⟨float⟩	Decimal	30.00

Table 5: Level script fields

Field	Meaning
background	Defines the stage background (bluesky is currently the only choice)
setting	Defines the stage setting (desert is currently the only choice); note that the setting surface is positioned at y=0
startx	The x position that the hero begins at
starty	The y position that the hero begins at
archpos	Position of the arch on the goal platform (left or right)
xleft	Sets x position of the left side of the object's bounding box
xmid	Sets x position of the x-axis center of the object's bounding box
xright	Sets x position of the right side of the object's bounding box
ybot	Sets y position of the bottom of the object's bounding box
ymid	Sets y position of the y-axis center of the object's bounding box
ytop	Sets y position of the top of the object's bounding box
thickness	Thickness of platform/goal (vertical thickness) or wall (horizontal thickness); defaults to 1.0
rot	Rotation of object about the z-axis (in degrees)
w	Moving platform width
speed	Moving platform speed (1-10)

5.2 Stage Section

The stage section begins with the header `**stage**` and defines basic information about the stage. The stage section has the following form:

```
**stage**
background=bluesky
setting=desert
startx=<float>
starty=<float>
```

This section is required.

5.3 Goal Section

The goal section begins with the header `**goal**` and defines the location of the goal. The goal has the following form:

```
**goal**
x=<float> to <float>, {ybot|ymid|ytop}=<float>[, thickness=<float>]
archpos={left|right}
```

Example:

```
**goal**
x=45 to 60, ytop=22, thickness=22
archpos=left
```

Note that the x range should be specified from left to right.

5.4 Platforms Section (Optional)

The platforms section begins with the header `**platforms**` and defines the platforms contained in the stage. The following syntax is used to define platforms:

```
x=<float> to <float>, {ybot|ymid|ytop}=<float>[, thickness=<float>]
```

Example:

```
**platforms**
x=45 to 60, ytop=22, thickness=22
x=-80 to -60, ybot=55
x=-45 to 0, ymid=78.5
```

Note that the x range should be specified from left to right.

5.5 Walls Section (Optional)

The walls section begins with the header `**walls**` and defines the walls contained in the stage. The following syntax is used to define walls:

```
{xleft|xmid|xright}=<float>, y=<float> to <float>[, thickness=<float>]
```

Example:

```
**walls**
xleft=20, y=22 to 40, thickness=10
xmid=80, y=0 to 160
```

Note that the y range should be specified from bottom to top.

5.6 Ramps Section (Optional)

The ramps section begins with the header `**ramps**` and defines the ramps contained in the stage. The following syntax is used to define ramps:

```
x=<float> to <float>, {ybot|ymid|ytop}=<float> to <float>
```

Example:

```
**ramps**
x=20 to 50, ytop=22 to 30
x=500.1 to 145.7, ybot=0 to 160
```

Note that the x range should be specified from left to right and the y range should be specified from bottom to top.

5.7 Moving Platforms Section (Optional)

The moving platforms section begins with the header `**movingplatforms**` and defines the moving platforms contained in the stage. The following syntax is used to define moving platforms:

```
w=<float>, speed=<float>
    {xleft|xmid|xright}=<float>, {ybot|ymid|ytop}=<float>
    ...
    {xleft|xmid|xright}=<float>, {ybot|ymid|ytop}=<float>
    end
```

Example:

```
**movingplatforms**
w=15, speed=5
    xleft=8, ybot=50.75
    xleft=8, ybot=10
    xmid=99, ytop=67
    end
w=5, speed=7
    xleft=66, ybot=100
    xleft=44, ybot=110
    end
```

Note that the number of nodes that define the path is unlimited.

5.8 Hazards Section (Optional)

The hazards section begins with the header `**hazards**` and defines the hazards contained in the stage. The following syntax is used to define hazards:

```
    spikes, {xleft|xmid|xright}=<float>, {ybot|ymid|ytop}=<float>[,
rot=<float>]
    spear, {xleft|xmid|xright}=<float>, {ybot|ymid|ytop}=<float>[,
rot=<float>]
    boulder, {xleft|xmid|xright}=<float>, {ybot|ymid|ytop}=<float>
```

Example:

```
**hazards**
spikes, xleft=4.0, ytop=19.0
spear, xmid=18.0, ymid=8.5, rot=270.0
boulder, xright=-44.0, ybot=60
```

6 FAQ

6.0.1 *Can I add more than three stages?*

No, the game supports a maximum of three stages. Stages 1-3 must be modified if you want to create your own stages.

7 Troubleshooting

7.1 Compilation Errors

7.1.1 *CMake fails to generate build files.*

Ensure that you have **gcc** installed. If this is not the issue, refer to the documentation provided at cmake.org for further troubleshooting.

7.1.2 *The compilation process fails while running the makefile.*

The makefile generated by **CMake** provides detailed error reporting. In most cases failure results from a missing function or library. Instructions for resolving a particular issue can usually be found via an online search of the error message.

7.1.3 *CMake is using an older/wrong version of* gcc *or defaulting to a different compiler.*

If you are using Windows, ensure that the desired version of **gcc** appears before all others in your PATH variable. If a different compiler is being used by **CMake**, double check that you are inputting the correct command (the `MinGW Makefiles` generator should default to **gcc**).

If you are using OS X or Linux, you can explicitly select the compiler by exporting the CC and CXX environment variable prior to calling **CMake** using the following commands:

```
export CC = path_to_gcc
export CXX = path_to_g++
```

7.1.4 *I am trying to use a compiler that is not gcc with the installer and it is not working.*

gcc is the only officially supported compiler at this time. Other compilers may or may not work with the provided installer and their use is not advised. If you are determined to use a different compiler, manual installation is recommended. Modifications to the source code of the game and/or the libraries may be required.

7.2 Runtime Errors

7.2.1 *The game creates a terminal/command prompt window which immediately closes.*

This indicates that the game has encountered a startup error. Try running the game from the terminal/command prompt so that any error messages produced before the game exits remain visible.

7.2.2 *The game encounters an error while loading an object.*

Check that the data folder in the `source` directory was copied into the `game` directory.

7.2.3 *The game encounters an error while loading a texture.*

Check that the data folder in the `source` directory was copied into the `game` directory.

7.2.4 *The game encounters an error while loading a sound file.*

Check that the data folder in the `source` directory was copied into the `game` directory.

7.2.5 *The game encounters an error while loading shaders and complains about an unsupported GLSL version.*

GLSL v330 is required to run the game. This message indicates that your system does not support OpenGL v3.3.

7.2.6 *I am using a system with a GPU that supports OpenGL v3.3 or higher but the game does not run.*

Make sure that your GPU drivers are up to date.

7.2.7 *I am using a notebook that supports OpenGL v3.3 or higher but the game does not run.*

This can happen if you are using a notebook computer with both an integrated and a dedicated GPU. Your notebook may be using the integrated GPU by default, in which case you will need to explicitly tell your system to run the game executable with the dedicated GPU. Instructions for doing this vary and are beyond the scope of this guide.