User Documentation
for
The Walking Game

Star Team
Version: 1.0
Date: 11/14/2014

Revision History

Revision #	Date	Description	Author
1.0	10/31/2014	Initial revision	Samuel I. Gunadi

Contents

1 Definitions, acronyms, and abbreviations

OpenGL — Open Graphics Library

OBJ — Wavefront OBJ File Format

GLEW — The OpenGL Extension Wrangler Library

PDF — Portable Document Format

IDE — Integrated Development Environment

MMORPG — Massively Multiplayer Online Role-Playing Game

2 Introduction

The members of Star team are Samuel I. Gunadi, Roberto J. Kondurura, and Ryan Elegant. The project website can be found at `https://github.com/samuelgunadi/thewalkinggame`.

The Walking Game features a Wavefront OBJ file parser; people 3D models, textures, and walking animations; textures with alpha blending; a textured floor; collision detection; simple command interpreter and scripting; nameplates; and an MMORPG style third-person camera.

The animation technique used in The Walking Games will not be skeletal animation; instead, it will be by loading one .obj for every key frame. This results in very large files.

3 User interfaces

The Walking Game includes an interface resembling an RPG game with a console to provide additional functionalities and aid in testing and debugging process.

4 Hardware requirements

The Walking Game should run on any computer hardware meeting the following criteria:

- Dual-core CPU
- OpenGL 4.3 compatible GPU with 2 GB memory
- 2 GB free hard disk space.
- Mouse with 3 buttons
- Keyboard with US layout

- 2 GB of RAM

5 Character control

Input	**Action**
Esc	Exit the program
W, S	Move forward/backward
A, D	Turn left/right
Left, right	Pan camera
Up, down	Tilt camera
Page down, page up	Dolly camera
Mouse wheel	Dolly camera
Left click and drag	Pan and tilt camera
Right click and drag	Pan and tilt camera and turn

6 Console commands

Available commands:

```
help
exit
showallnames
createnpc [string:name] [int:model_id]
createnpc [string:name] [int:model_id] [float:x] [float:y] [float:z]
deletenpc [string:name]
setmodel [string:name] [int:model_id]
getmodelinfo [string:name]
setwalkingspeed [string:name] [float:speed]
getwalkingspeed [string:name]
setposition [string:name] [float:x] [float:y] [float:z]
getposition [string:name]
setangle [string:name] [float:angle]
getangle [string:name]
setanimationstate [string:name] [int:state]
getanimationstate [string:name]
takecontrol [string:name]
getcamerainfo
setcamerapos [float:x] [float:y] [float:z]
setcameratarget [float:x] [float:y] [float:z]
setcameraradius [float:radius]
```

```
getlightinfo
setlightpos [float:x] [float:y] [float:z]
setlightintensity [float:r] [float:g] [float:b]
setfpslimit [int:fps]
sleep [double:seconds] -- pause (for scripting)
loadscript [string:path] -- the file must be in ASCII encoding
    (sample script can be found in /binary/testscript.txt)
test -- for testing purposes
```

NOTE—All strings must not contain spaces.

7 Third-party libraries

The Walking Game integrates several external software to provide functionality:

1. GLFW. GLFW is used for creating windows with OpenGL contexts and receiving input and events. GLFW is multi-platform and supports Windows, OS X and many Unix-like systems. GLFW is licensed under the zlib/libpng license.

2. GLEW. GLEW is a cross-platform open-source C/C++ extension loading library. GLEW provides efficient run-time mechanisms for determining which OpenGL extensions are supported on the target platform. OpenGL core and extension functionality is exposed in a single header file. GLEW has been tested on a variety of operating systems, including Windows, Linux, Mac OS X, FreeBSD, Irix, and Solaris.

3. GLM. GLM is a header only C++ mathematics library for graphics software based on the OpenGL Shading Language (GLSL) specification and released under the MIT license.

Appendix A: Screenshots

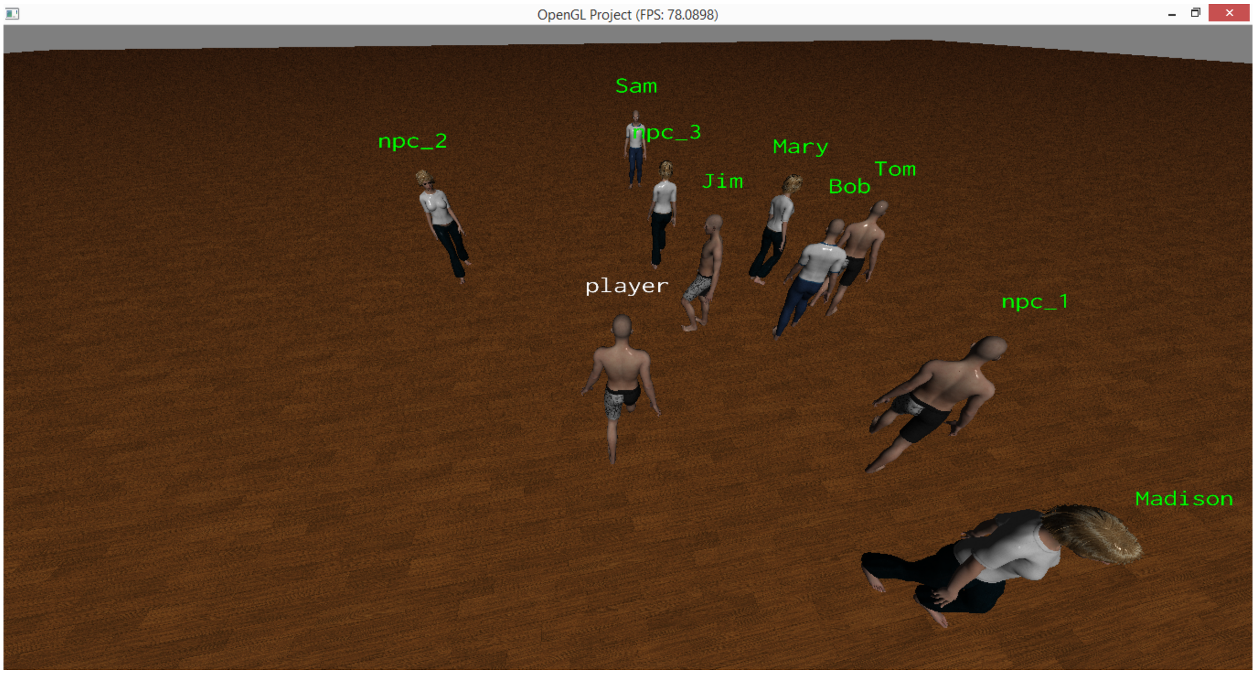
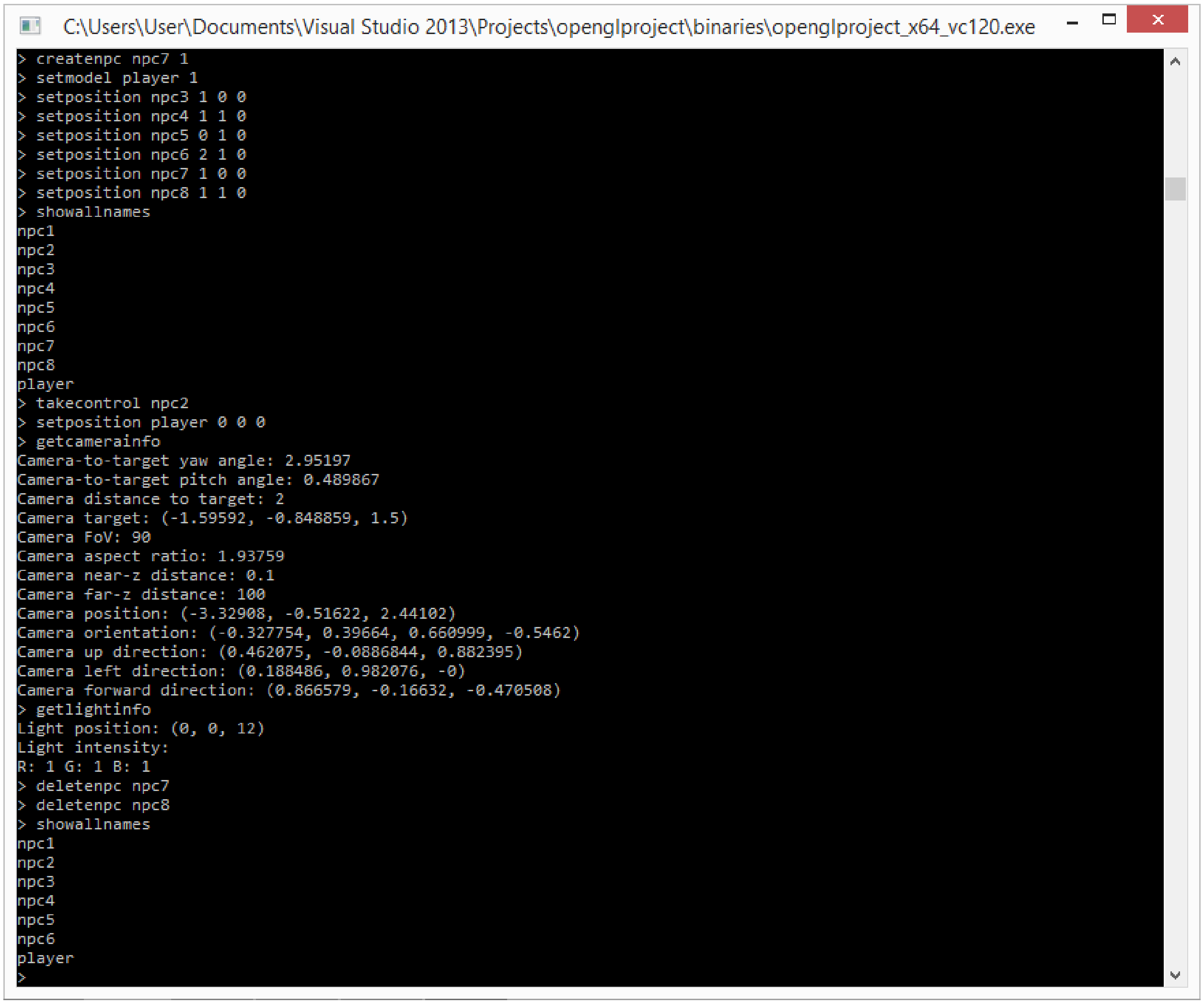


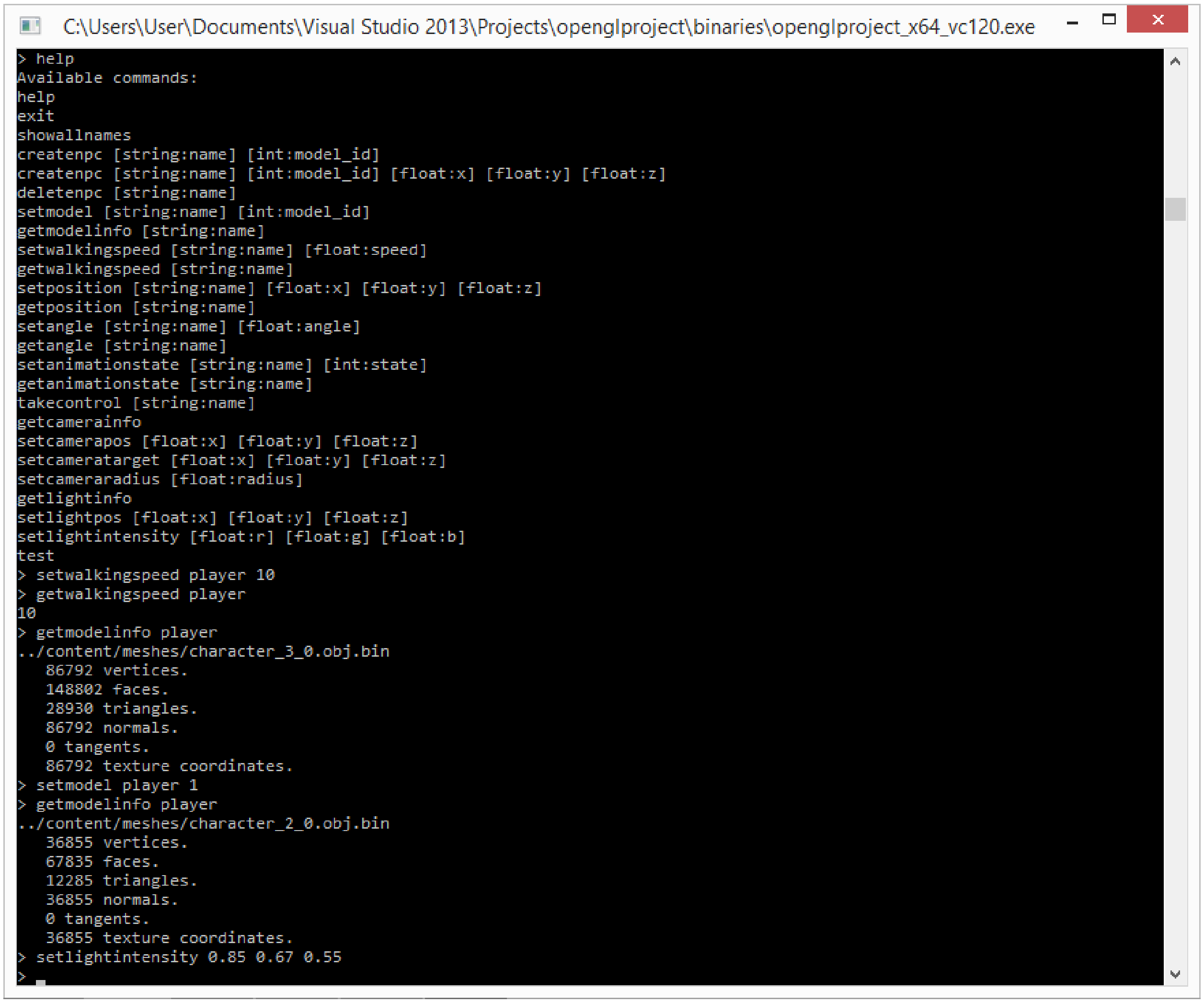
Figure 1: Main window of The Walking Game



C:\Users\User\Documents\Visual Studio 2013\Projects\openglproject\binaries\openglproject_x64_vc120.exe

```
> createnpc npc7 1
> setmodel player 1
> setposition npc3 1 0 0
> setposition npc4 1 1 0
> setposition npc5 0 1 0
> setposition npc6 2 1 0
> setposition npc7 1 0 0
> setposition npc8 1 1 0
> showallnames
npc1
npc2
npc3
npc4
npc5
npc6
npc7
npc8
player
> takecontrol npc2
> setposition player 0 0 0
> getcamerainfo
Camera-to-target yaw angle: 2.95197
Camera-to-target pitch angle: 0.489867
Camera distance to target: 2
Camera target: (-1.59592, -0.848859, 1.5)
Camera FoV: 90
Camera aspect ratio: 1.93759
Camera near-z distance: 0.1
Camera far-z distance: 100
Camera position: (-3.32908, -0.51622, 2.44102)
Camera orientation: (-0.327754, 0.39664, 0.660999, -0.5462)
Camera up direction: (0.462075, -0.0886844, 0.882395)
Camera left direction: (0.188486, 0.982076, -0)
Camera forward direction: (0.866579, -0.16632, -0.470508)
> getlightinfo
Light position: (0, 0, 12)
Light intensity:
R: 1 G: 1 B: 1
> deletenpc npc7
> deletenpc npc8
> showallnames
npc1
npc2
npc3
npc4
npc5
npc6
player
>
```

Figure 2: Console commands example #1


C:\Users\User\Documents\Visual Studio 2013\Projects\openglproject\binaries\openglproject_x64_vc120.exe

```
> help
Available commands:
help
exit
showallnames
createnpc [string:name] [int:model_id]
createnpc [string:name] [int:model_id] [float:x] [float:y] [float:z]
deletenpc [string:name]
setmodel [string:name] [int:model_id]
getmodelinfo [string:name]
setwalkingspeed [string:name] [float:speed]
getwalkingspeed [string:name]
setposition [string:name] [float:x] [float:y] [float:z]
getposition [string:name]
setangle [string:name] [float:angle]
getangle [string:name]
setanimationstate [string:name] [int:state]
getanimationstate [string:name]
takecontrol [string:name]
getcamerainfo
setcamerapos [float:x] [float:y] [float:z]
setcameratarget [float:x] [float:y] [float:z]
setcameraradius [float:radius]
getlightinfo
setlightpos [float:x] [float:y] [float:z]
setlightintensity [float:r] [float:g] [float:b]
test
> setwalkingspeed player 10
> getwalkingspeed player
10
> getmodelinfo player
../content/meshes/character_3_0.obj.bin
   86792 vertices.
   148802 faces.
   28930 triangles.
   86792 normals.
   0 tangents.
   86792 texture coordinates.
> setmodel player 1
> getmodelinfo player
../content/meshes/character_2_0.obj.bin
   36855 vertices.
   67835 faces.
   12285 triangles.
   36855 normals.
   0 tangents.
   36855 texture coordinates.
> setlightintensity 0.85 0.67 0.55
> _
```

Figure 3: Console commands example #2

References

[1] "Systems and software engineering – developing user documentation in an agile environment," *ISO/IEC/IEEE 26515 First edition 2011-12-01; Corrected version 2012-03-15*, pp. 1–36, March 2012.

[2] P. Tripathy and K. Naik, *Software Evolution and Maintenance: A Practitioner's Approach*. Hoboken, New Jersey: John Wiley & Sons, 2014.

[3] K. Naik and P. Tripathy, *Software Testing and Quality Assurance: Theory and Practice*. Hoboken, New Jersey: John Wiley & Sons, 2008.

[4] I. Sommerville, *Software Engineering*, 9th ed. Boston, Massachusetts: Addison-Wesley, 2010.

[5] R. Pressman and B. Maxim, *Software Engineering: A Practitioner's Approach*, 8th ed. New York, New York: McGraw-Hill Education, 2014.

[6] K. M. Lui and K. C. C. Chan, *Software Development Rhythms: Harmonizing Agile Practices for Synergy*. Hoboken, New Jersey: John Wiley & Sons, 2008.

[7] J. Gregory, *Game Engine Architecture*, 2nd ed. Boca Raton, Florida: CRC Press, 2014.

[8] P. Shirley and M. Ashikhmin, *Fundamentals of Computer Graphics*, 3rd ed. Natick, Massachusetts: A K Peters, 2009.

[9] J. F. Hughes, A. v. Dam, M. McGuire, D. F. Sklar, J. D. Foley, S. K. Feiner, and K. Akeley, *Computer Graphics: Principles and Practice*, 3rd ed. Boston, Massachusetts: Addison-Wesley, 2013.

[10] E. Angel and D. Shreiner, *Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL*, 6th ed. Boston, Massachusetts: Addison-Wesley, 2011.

[11] S. J. Janke, *Mathematical Structures for Computer Graphics*. Hoboken, New Jersey: John Wiley & Sons, 2013.