[Document title]

[Document subtitle]

Hancox, Liam

[Year]

Contents

[Introduction 2](#_Toc197944194)

[3D Model Rendering 2](#_Toc197944195)

[Texture Mapping 2](#_Toc197944196)

[Lighting 2](#_Toc197944197)

[Transparency 2](#_Toc197944198)

[Cameras 2](#_Toc197944199)

[Interaction 2](#_Toc197944200)

[Other Aspects 2](#_Toc197944201)

[Referencing 3](#_Toc197944202)

# Introduction

This document will investigate the design and implementation of various methods used in the creation of a clone version of Bullfrog’s *Dungeon Keeper 2* title. For each of the key elements, research and analyse has been carried out looking into how the original title has achieved them and then comparing these to other related titles. Then looking into how each of these elements have been implemented into the clone version created for this project and if/how they can be used as part of the wider project.

# 3D Model Rendering

The Dungeon Keeper 2 title used the Graphics API Direct3D (PCGamingWiki, 2024) as they wanted to make a 3D game that utilised hardware acceleration. This was implemented to upgrade and replace the original titles pre-rendered sprites allowing the game world and its inhabitants to be a 3D/2D hybrid (Wiki, 2025). The OpenGL graphics API (KhronosGroup, 2025) was used to achieve this, a collection of OBJ files were loaded through a manifest via a model factory, where an ASSIMP (Open Asset Import Library) library parses the file and extracts all the relevant information such as vertex positions and normal etc and this data is then uploaded to OpenGL via vertex buffer objects (VBOs), vertex array objects (VAOs) and an index buffer to improve render speed. Once all objects are loaded a render function is called which iterates over all the game objects that have been loaded from the manifest and draws the objects to screen.

# Texture Mapping

Dungeon Keeper 2 makes use of many textures, including varying textures across walls and terrain. An assumption is made that each terrain/wall type has its associated predefined group of textures and during generation the game randomly selects from the appropriate group for each instance to avoid repetition. When a ExampleGO is textured in this application, it is loaded from the manifest and reads which texture is to be applied to it as it makes a call and creates a pointer to the AIModel class allowing access to the , then inside of the PreRender function it is assigned it’s appropriate texture units in the Initialisation, the main texture (GL\_TEXTURE0) and it if has the shader for doing so attach the normal map too (GL\_TEXTURE1).

# Lighting

# Transparency

# Cameras

# Interaction

# Other Aspects

# Referencing

PCGamingWiki (2024) *Dungeon keeper 2*, *Dungeon Keeper 2 - PCGamingWiki PCGW - bugs, fixes, crashes, mods, guides and improvements for every PC game*. Available at: https://www.pcgamingwiki.com/wiki/Dungeon\_Keeper\_2#API (Accessed: 12 May 2025).

KhronosGroup (2025) *The industry’s foundation for High Performance Graphics*, *OpenGL.org*. Available at: https://www.opengl.org/ (Accessed: 12 May 2025).

Wiki, C. to D.K. (2025) *Dungeon keeper 2*, *Dungeon Keeper Wiki*. Available at: https://dungeonkeeper.fandom.com/wiki/Dungeon\_Keeper\_2 (Accessed: 12 May 2025).