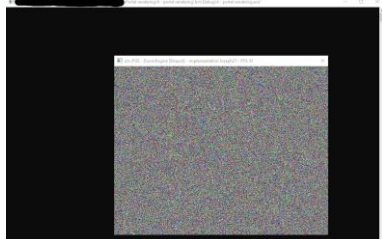
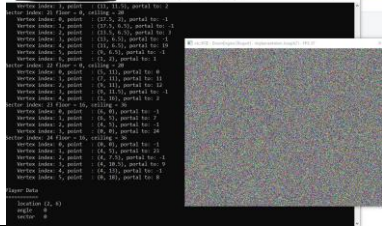
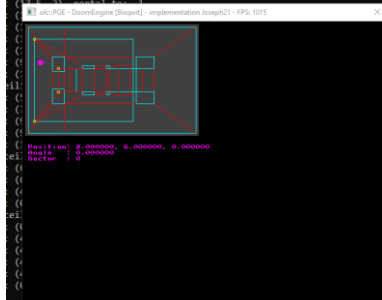
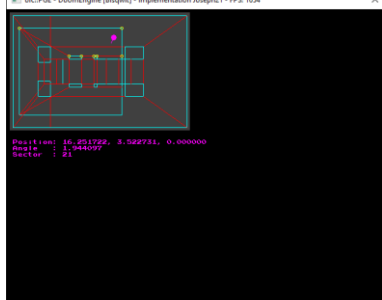
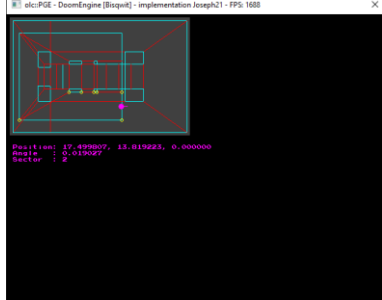


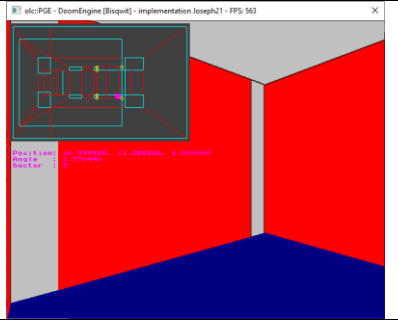
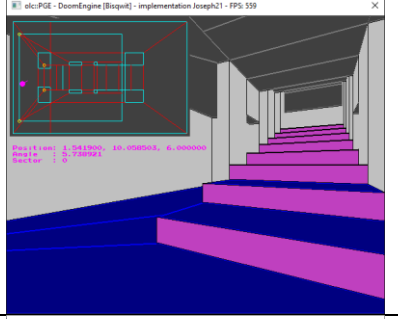
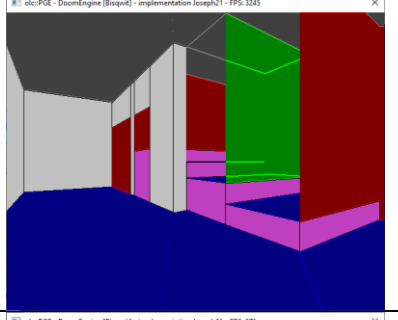
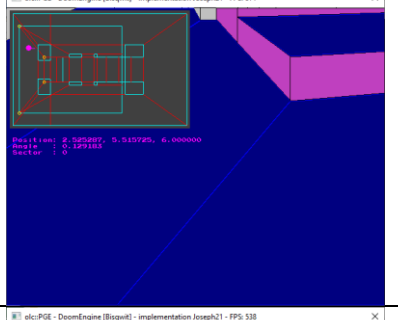
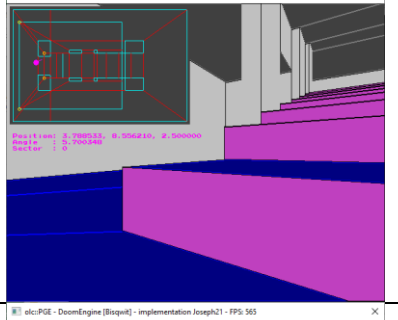
Portal rendering implementations series

Based on Bisqwit video (<https://youtu.be/HQYsFshbkYw>)

Joseph21, August 15, 2022

All source files on: <https://github.com/Joseph21-6147/Portal-rendering>

Nr	Source file name	Subject	Preview
1a	main - part 1a - setup PGE, datastructures.cpp	This part comprises only the setup of the PGE skeleton program and the definition of the Sector and Player data structures needed for the portal rendering.	
1b	main - part 1b - Load- and UnloadData.cpp	Implementation of data input and parsing into the data structures	
2a	main - part 2a - map and player drawing (2D).cpp	Added: 2D map drawing, with player and direction indicator - Just the drawing, there's no player movement yet.	
2b	main - part 2b - MovePlayer() and user interaction.cpp	Player (horizontal) movement (forward, backward, strafing and rotation). No collision detection yet.	
2c	main - part 2c - horizontal collision detection.cpp	Collision detection for the horizontal player movements, to prevent walking through walls, stairs, etc.	

Nr	Source file name	Subject	Preview
3a	main - part 3a - initial portal rendering.cpp	Added: the first (initial) and limited form of 3D portal rendering – adjacent portals are displayed as placeholders.	
3b	main - part 3b - incremental portal rendering.cpp	In this implementation the adjacent portals are rendered as well, using a queue and looping through it until it's empty.	
3c	main - part 3c - slomo demo version of rendering algorithm.cpp	This is a special implementation (not part of the progressive series), that is created to illustrate (in slow motion) how the rendering algorithm behaves per frame.	
4a	main - part 4a - looking up and down.cpp	NOTE: This implementation elaborates on <u>part 3b</u> , not part 3c! This part is a limited adaptation of 3b, and implements mouse aiming and vertical looking.	
4b	main - part 4b - jumping, crouching, vertical collision detection.cpp	Where horizontal movement was introduced in part 2b, and vertical looking in part 4a, this part introduces vertical movement: jumping and crouching. It also implements the collision detection that is needed to keep the player between the ceiling and the floor.	
4c	main - part 4c - simple distance shading.cpp	This final part implements a simple form of distance shading as the finishing touch of the series.	