Keksimil🍪

(Bosnian: keks i mlijeko -> cookies and milk)

Basic info

Technology: C# with ImGui or Avalonia or Eto.Forms
Scope: definitely large
Licence: MIT
Target audiences: me, Half-Life mappers, BurekTech X mappers
Target HW and SW: Windows 7+, Linux, DirectX 12 or Vulkan (by the time this is done, nobody will be on 2010 hardware, so)
Idea: plugin-rich map editor

Features

Plugin interfaces for:

- model formats *(OBJ, FBX, MDL)*
- texture formats *(WAD, PNG)*
- map formats *(importing and exporting; MAP, RMF, VMF)*
- filesystem extensions *(if game assets are in a .pak file)*
- render frontend extensions *(lightmap preview, entity visualisations etc.)*
- misc utilities *(VHLT GUI frontend, VHLT portalfile and pointfile parser)*
- node type plugins *(pipe node type, heightmap terrain node etc.)*
    - node types may have custom editors/tools
- definition plugins *(FGD, Doom 3 DEF)*
- miscellaneous utilities *(new entries in File and/or Edit tabs)*

Most basic, built-in functionality without any plugins:

- practically none, as there are basically no tools
- viewports
- native map project format: “KSM”
- moving around in the 3D viewport
- prefab library (could be a plugin?)
- reloading of **any** asset

There would be 2 types of configurations: engine configurations and game configurations.
Engine configurations would handle:

- Compiler presets
- Collections of plugins (what map format to use, what texture formats to use etc.)

Game configurations would handle:

- Entity definitions

- Game and mod directories
- Collections of textures

Controls for navigation will be in the style of TrenchBroom, Unity etc.
Controls for manipulating objects will be in the style of Blender, generally referring to the GSR scheme.

Base plugins

- Basic brush nodes with a cylinder and arch node too
- Basic mesh nodes with a few presets: subdivided plane, cube, cylinder etc.
- Basic texture formats: PNG, JPG, TGA
- Basic model formats: OBJ, glTF2
- Basic import/export formats: OBJ, .map (Valve220)

Other features, some of which could be plugins, some built-in:

- Non-destructive workflow
- Negative brushes
- Reference images
- Pipe node type
- Terrain node type
- Unique node IDs
- Layers
- Visibility categories (e.g. "Hide NULL" or "Hide func_detail")
- Surface properties
- UV editor
- Check for problems widget / issue generators

Development

Ideas to prototype:

- Untextured brush & mesh editing in 3D
    - Requires mini rendering code
    - Requires mini UI & input
- Node type plugins
    - Requires a plugin system
    - Node types will need to support decomposition, i.e. when exported into a non-native format (.map), they should be able to decompose into brushes or triangle meshes (OBJ)
    - In the native map format, each node will write its own data
- Node tree
    - Requires matrix & parent-child maths
- Renderer extension / viewport plugins
    - Each viewport type could be its own plugin essentially (e.g. HL lightmapped preview, textured shaded, wireframe, set of 2D viewports)

    - Must handle input somehow. 3D viewports obviously support raycast picking and frustum (rectangle in the user's eye) selection, but… you could frankly say the same for 2D viewports, so this is probably already solved
- Texture & material plugins
- Entity properties, groups etc.
- Entity definition format plugins

Estimated time to make the first couple of dirty prototypes: a month or two.
Estimated time to reach the 1st sort of milestone: 18 months.