* Essential:
  + ~~Set up version control~~
  + ~~Handle Text~~
  + ~~Convert Player to Entity~~
  + Entity Component System
  + ~~Input Handler~~
  + Fix abstract and unknowable framerate issues.
    - Try seeing if disabling shadows fixes framerate. See if hardware accelerated.
      * Disabling draw\_circle\_full shadows improved FPS tremendously. Function is definitely slow. Try to see how you can make it faster or replace entirely.
      * Replace circle shadows with colour mod blended black surfaces that are skewed and warped to add depth.
  + Refine collisions and add sliding along walls, if possible.
    - There is some way of clipping through walls which I have no doubt I will see again. Currently no idea how to replicate but it’s when the very edge of your box is along the seam between walls. Try to do something about it.
    - Collision happens when you are already colliding on one edge and the pre-movement box collides with a third object, possibly having enough delay time-wise for a single frame of uninterrupted forward movement to get through.
    - This also happens whenever you align on the seams of blocks, even with a larger collider box. I am hoping that the multiple reference set object I made will help with its multiple checks of unique data.
    - Collisions work as intended only when going diagonally up-right and down-left, the other axis also exhibit the previous weird behaviour. The new wall pre-colliders help, but somehow don’t eliminate the overstep in collision.
  + Find a way to reference the event polling for single key down in the main loop.
    - Basic implementation of this is done, but it still checks continuous input for some reason.
    - Throws an exception when modifier keys such as Shift and Control are pressed.
  + Add a Restart button or function. Bind to R.
  + Make a container class for Wall and other interactable objects, so that they can be drawn at once and moved accordingly. Call it Building or something like that.
  + Debug text is having weird properties on FPS counter when debug mode is active – translated right and random 0 appears in the middle of the screen.
  + ~~Camera movement and viewport.~~
  + ~~Basic Wall~~
  + ~~Basic Wall Collisions~~
  + Figure out elegant way to rescale text on demand without reloading font.
  + Interactivity Context Menu
  + Set up a scalable storage object that you can use to sort entities based on y-axis depth as a bootleg depth buffer before bliting to screen.
  + The UI text is disappearing and is also flickering whenever it’s the first or last element drawn, this is, I imagine, because of the original skeleton I use to copy the clones from, which I cannot delete, because it wasn’t generated at runtime. Consider moving the skeleton system to dynamic allocation to fix this problem.
* Achievable:
  + Basic TCP Networking
  + Rebindable keys
* Pipe dream:
  + AI with pathfinding
  + Procedural Dialogue System, that doesn’t feel like shit.
  + Procedural Quest System, that doesn’t feel like shit.

Pitch:

A game where most objects are interactable in a modular way that allows you to achieve a higher level of immersion by allowing you to sometimes do functionally pointless options, but which allow for a greater level of roleplay and character development.

Entities and objects are to be constructed int such a way as to be modular and easily updatable, so that if no interactions exist in the moment, they can be added later to a broad selection of entities and objects.

In genre terms, the game should feel like a single-player MMORPG, where the player has a vast world to explore and progress through – some parts procedurally generated, others hand-made. In reference to other existing titles, it will incorporate systems similar the procedural modularity of Dwarf Fortress’ Adventure mode, Old-School Runescape’s skill levelling system, Morrowind’s hyperlink-based dialogue system, and the general plot outline and quest structure of Fallout: New Vegas.

Narratively the game contextualises itself as a bounty hunting game, where you are given one procedurally generated target somewhere in the world with no concrete information on where they are or who exactly they are. As the player converses with NPCs and completes minor tasks throughout the world and the various settlements within it, they uncover more about the target. In order to make the progression more compelling, the deductions are based on excluding misleading information, rather than directly receiving an answer. The player is to make up their own reasoning and decide what to trust and what to not.

The player will ideally have several options when confronting the target, ranging from killing them on the spot, sparing them and leaving them alone, taking them in alive, or abandoning the quest entirely and living a somewhat simulationist experience.

The game should be designed around the idea of hardcore permadeath, but can optionally have a mode with very sparse checkpoints. An alternative softcore mode can be essentially a somewhat mocking version where you are either an immortal character who cannot be killed and this is mentioned via additional dialogue as negative with all of the suicidal options are disabled and are replaced with funny quips and the player is nudged towards the regular hardcore mode for the “full” experience, since the softcore version will end up mercifully easy when consequences aren’t there.

* When creating all entities, make a singular sprite SDL Surface for them to reference at creation instead of having every single entity initialise sprites and textures for themselves. Animation might make that change, but for now keep it in mind.

The Character Generator:

The character generator will select from an initially shallow but ultimately modular enough to expand pool of facial features and other biographical data to create a large number of characters to populate the world with.

The currently planned list of features are things such as first name, surname, place of birth, hair colour, hair style, nose, eye colour, markings, and tattoos, spoken language(s),

NPCs will also likely have a last known location which will be updated based on their proximity to other NPCs. This can perhaps be modified by certain traits or character quirks which will probably play some part in the character system.

Traits can be things like Elusive, Charismatic, Stinking, Polite, etc. Elusive for instance will reduce the chance of them being notices, so that when asked NPCs are less likely to have an answer to questions about the character’s last known location. To counter such overtly strong skills, NPCs can have traits such as Keen Eyed, which will have the opposite effect. It would be interesting to play around with the idea of traits such as Strong Long-Term Memory, Weak Short-Term Memory, or vice-versa. Seeing as a bounty might have passed through a town several days ago and some NPCs might not remember anymore. Traits such as Self-Aware might allow NPCs to reveal in dialogue some of their traits without necessarily having a UI element listing them all, though not all NPCs will be so introspective.

The concept of traits having side-grades or inverse traits to counter them or enhance them is an interesting idea, which I should play with.

The World Generator:

The current plan is to either have a completely hand made world with no randomisation or make hand-made chunks with which to generate a decently large landmass. If it turns out that solutions to the bounty hunter mode are too predictable, there is room for a completely random mode, which would procedurally generate a desired world size, which can be added post-launch or at a later time in development if ahead of schedule.