# Methodology

* 1. **Basic play information**

The game developed in this project is solely based on PC attached with commodity VR set. For gameplay integrity, it is essential that any Oculus model be used, combined with dual motion controllers. For optimal runtime performance, the recommended hardware setup includes but is not limited to:

* GPU: GTX 1070 or higher equivalent
* RAM: 16GB or higher
* CPU:  Intel Core i7 or higher equivalent
* OS: Windows 10

The playing stance may be either standing or seated, while the latter is recommended. Players may apply any horizontal body orientation and rotate any time during play. Players may also move physically within safety capacity of VR guard zone (if configured properly).

For health concern, players with motion sickness should play with caution, and must quit playing as soon as possible when experiencing obvious dizziness. In addition, the game content may be harmful to players ever confirmed with (not limited to) following symptoms:

* Photosensitive epilepsy
* Acrophobia
* Claustrophobia
* Thalassophobia
* Megalophobia

Players with any diseases above should avoid playing the game for safety.

The game is free of following contents:

* Strong language
* Violence
* Sex indication
* Use of drugs, tobacco or alcohol
* Thrill or occult phenomenon
  1. **Game flow**

As a technical verification prototype, no complete game flows are presented in the game. There are no explicit guidance or goals directing players’ gaming conduct. During user test, players will be encouraged to explore existing mechanisms and objects in environment.

* 1. **Player hierarchy**

The player character’s component hierarchy was inherited from UE4 VR template. The root component, as well as VR origin transform, represents center of playzone in reality. The camera attached to VR origin is mastered by VR head-mounted display, where player’s view is based. One pair of hands will be spawn at runtime and attached to VR origin, mastered by paired VR motion controllers. As is designed to imitate head stance, HUD is directly attached to camera to maintain relative transform with simplicity.

A mobile node is attached to VR origin. The node has its position and yaw imitating camera (i.e. HMD) while maintaining its relative pitch and roll. This particular design emphasizes player’s body position and orientation in reality, as reflected by the capsule collider attached to it.

* 1. **HUD**

Inspired by realistic head-up displays, HUD in the game simulates intuitive, minimalized holographic projection on the astronaut mask. As a feedback segment of human-computer interaction, the HUD provides visualized essential information on system state, including motion state, equipment status, and spatial awareness (see figure 1.1).

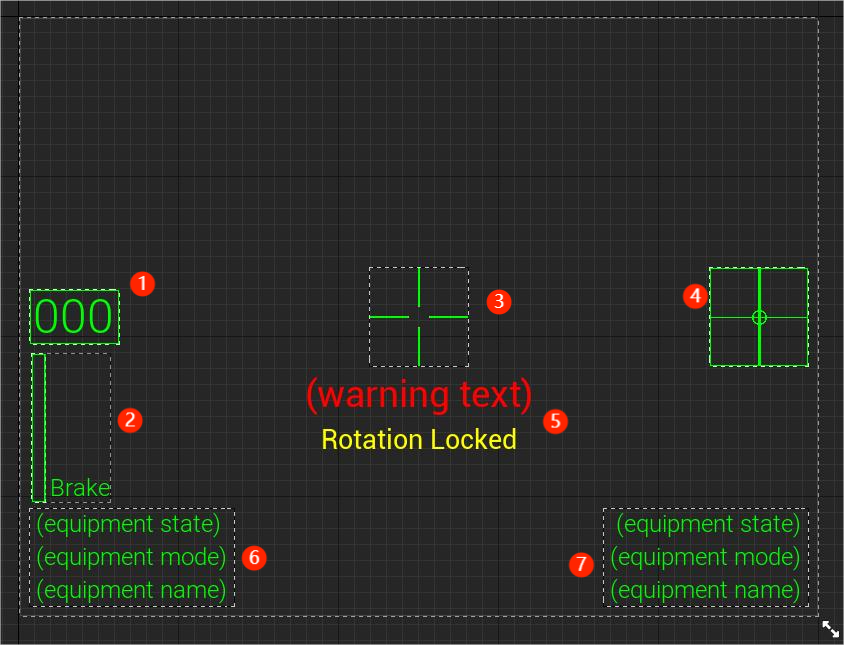


figure .: HUD in widget design view.

Legend:

1. Speed indicator. Unit is meter per second.
2. Brake strength indicator. Only present when brake input is greater than 0. Progressive bar shows current brake input scale.
3. Fixed crosshair.
4. Rotation input indicator. Only active when rotation is unlocked.
5. Warning texts. Only visible when corresponding condition applies.
6. Left hand equipment info panel.
7. Right hand equipment info panel.

Similar to real-life HUD, pure green (#00BF00) was chosen as theme color of the HUD. Studies suggested that human eyes are most sensitive to green colors and are best at distinguishing among them. As possible adaptation to natural life, human eyes also appeared to be less exhausted staring at green than other colors.

An attempted was made in early development stage to deliver HUD as conventional widget, a summary of Unreal Engine 2D entities. The outcome appeared to be complete failure in visual deliverance, as HUD elements were rendered at zero depth of field (referred as DoF below) on solely left VR goggle. With reference to multiple sources, this phenomenon was explained due to following UE4 rendering logics:

1. Widgets in UE4 are always rendered at post-processing stage, implying they may never participate in 3D rendering.
2. The VR binoculus (double-eyed) displays were conventionally divided into 2 separate screens (one per VR lens), with screen UVs respectively mapped to U value of 0-1 on left screen and 1-2 on the right. (The similar behavior also applies to dual screen setup based on conventional displays.) The default post-processing material of widgets only renders to screen UV of 0-1, namely all contents are delivered on the leftmost display.

The principles above implies the incapability of 2D widgets on presenting HUD in VR mode. It is also suggested by the sources that all visual items must be rendered in 3D world to form focusable, stereo vision in head-mounted display.

The methodology was adopted in second attempt, where all HUD elements are projected on 3D surfaces with constant focal distance of approximately 60 meters, rendered occluding other world entities. This particular method, inspired by the game VTOL VR, eventually reached proposer’s expectation.

The rendering logics above were also flexibly utilized for presenting monocular (single-eyed) vision, inspired by early Apache Attack Helicopter’s monocular eyepiece. Phasmophobia, a VR thrill game, is also known for utilizing the mechanism for strengthening thrilling atmosphere. As a potential extension to human-computer interaction, players may choose to keep a single eye open to switch between HUD overlaid view and HUD-free view accordingly. Nonetheless, it is a notable fact that monocular displays are being gradually replaced by binoculus ones in military utility, as they are noticed for causing discomfort in binocular vision.

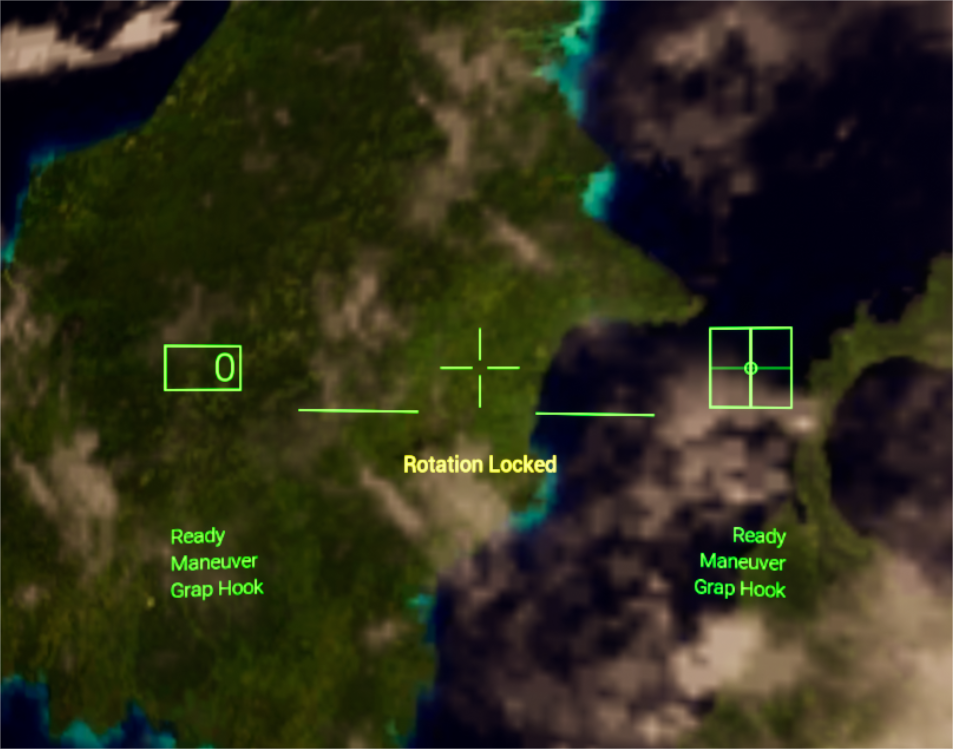


Figure .: HUD rendered in game.

(The screen capture is solely from HDM left eye and does not represent final effect in VR view.)

The HUD system also includes individual tracking marks attached to player character. As contribution to player’s spatial awareness, these marks were designed for marking orientations of any object or transform from player’s view. Possible objects include predicted hit point, interactive objects, mission objectives (see figure 1.3 below).

地图

描述已自动生成

Figure .: HUD with a track mark.

The green square with text ‘Locked’ marks an interactive power cell in the scene.

To form visual effect as HUD, location and rotation of all tracking marks are recalculated at real time, maintaining their focal distance on the extension line between player viewport and the tracked object. The marks are kept facing player viewport, meanwhile imitating tilting state (i.e. roll) of HMD – this is helpful when texts are embedded in HUD system, since the texts must be maintained horizontal to player’s view for optimal readability.

As a backup method of preventing HUD vision blockage, two virtual buttons were placed at forehead position of HMDs. Players can find the buttons by reaching their hands to front sides of HMDs, then pressing trigger, to quickly toggle HUD visibility.

* 1. **Jetpack**

Jetpack is designed as primary approach player moves as an astronaut in the game. Similar to real-life astronaut jetpacks, it provides 6 degree-of-freedom translational and rotation control by applying respectively linear and angular momentum to astronaut. In this game, differences are made where all controlling of the jetpack are assembled at a flight control stick located front left of player’s waist. This interaction model is supported by input set of Oculus motion controller, where sufficient inputs enabled rotation by rotating control stick, as well as locomotion using thumbsticks and buttons. The simplified interaction model also enabled one spare hand for complex tasks in the game.

Rotation control is simulated by rotation of motion controller, with controlling logic similar to real-life aircraft control sticks. Yaw, pitch and roll are mapped to corresponding rotation actions. At the start of game, rotation is locked to prevent accidental operation causing dizziness.

|  |  |
| --- | --- |
| Input | Action |
| Thumbstick Y | Move forward/backward |
| Thumbstick X | Move right/left |
| Y/B | Ascend |
| X/A | Descend |
| Trigger (axial) | Brake |
| Thumbstick press | Toggle rotation lock |
| Rotate around X axis | Pitch |
| Rotate around Y axis | Yaw |
| Rotate around Z axis | Roll |

Table .: Input set while motion controller is holding flight control stick.

* 1. **Grappling Hook**

Grappling hook is intended as secondary approach that player maneuvers in environment. It performs function by pulling player towards any position it hits. During development, the functionality was expanded to locking on specific object and pulling it towards player.

|  |  |
| --- | --- |
| Input | Action |
| Trigger | Fire |
| Y/B | Switch fire mode |

Table .: Input set while grappling hook is armed on one hand.

Logic of grappling hook launcher is based on state machine

Grappling hook projectiles is attached with engine’s projectile movement component, moving and triggering event based on physics. It also reacts differently according to target it hits. When hitting static scene

* 1. **Wrist menu**

Wrist menu acts as an integration of in-game configurations and commands.

* 1. **Tutorial**

In addition to core gameplay, a tutorial is set to instruct players with basic operations. The tutorial consists of a linear series of text guidance, each of which suggests player with one feature of core mechanism.

* 1. **Level design**

As complementary to the game mechanism and narrative, a zero-gravity space level is assembled for demo and user test. The level is placed with:

* Space station as the sole terrain mesh.
* Dummy celestial bodies – sun and earth, as background.
* Space skybox
* Volumes for