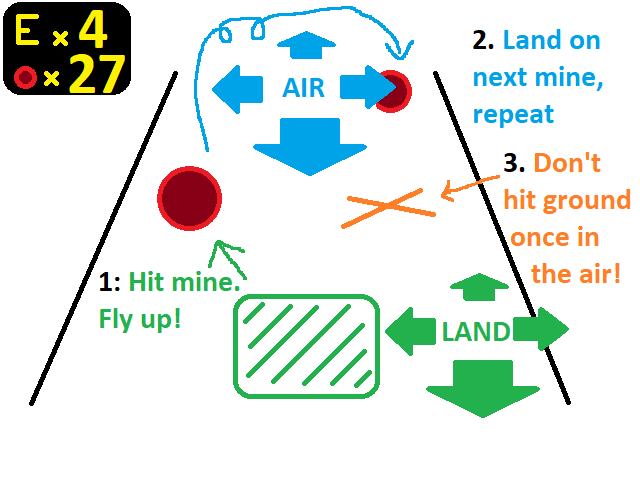
MVP SKETCH



PROJECT REQUIREMENTS: OOP PRINCIPLES:

* Abstraction:
  + Make use of functions
  + Mark example with comment (MastEr)
* Encapsulation:
  + Use accessors sensibly
  + Sanitize input using manually-written accessor functions when autogenerated accessors are insufficient
  + Mark example with comment (MastEr)
* Inheritance:
  + Create an abstract Mine class from which the different mines are derived
  + All mines have a point value and a move forward function
  + Mark example with comment (MineAbstract)
* Polymorphism
  + Most Mine class functionality would be virtual, if not abstract, since the mine class is not meant to be instantiated directly
  + Mark example with comment (MineAbstract)

PACKAGES:

* TMP Essentials – TextMeshPro UI

FEATURELIST

|  |  |
| --- | --- |
| Feature | Task |
| Loading Scene | * Game starts at loading scene * MastEr initialized |
| Menu  Scene | * Title Label * Top 3 high scores displayed * Name inputfield (if there was an old game manager, this is populated from player name upon clicking Return to Menu button; if not, it is taken from save file) * Start button * Reset highscores button (replaces scores with hardcoded, non-zero values, then resaves file) * Quit button * Overall, almost exactly like sceneflow project |
| Game Scene | * E-Jumps text * Score text * Return to Menu button |
| Player Control | * Move on ground with translate * Move in air by decrementing velocity, giving the impression of fighting against the vector and correcting it in the direction the player wants to move * Player must be bounded by invisible walls to prevent falling off the game plane * Need a player spawn platform so E-Jump isn’t used instantly onSpawn |
| Mine Spawning | * Spawn Manager instantiates mines of random types, at random times, in random locations * Danger and Ghost mines |
| Player-Mine Collision | * Player collisions with mines are detected * Collision adds mine-defined force to player * Collision adds mine-defined torque to player |
| Combo score | * Player collisions with mines add mine’s scoreValue to score * Score is reset if player collides with ground and E-jumps = 0 * Score displayed in onscreen UI * High score checked in MastEr. First time a score is updated, scoreChanged = true * If player collides with ground and E-Jumps = 0, check scoreChanged. If true, save data * High score is NOT displayed in onscreen UI. Instead, fire a bronze, silver, or gold particle effect if score > existing highscore. Display highscores only on menu screen, since player can continue playing if they hit the ground (no gameover screen) |
| E-jumps | * Player starts with 3 E-jumps * Player can get more by colliding with E-jump mines * If [player collides with ground | player presses spacebar] and E-jumps > 0, launch player upward and decrement E-jumps * E-jumps displayed in onscreen UI |
| Mine varieties | * Normal * Super – increased upward force * E-Jump – increment E-jumps. Same force as normal mine * Danger – unpredictable upward force, XZ force, and torque, making it difficult to coordinate the next mine impact. Worth substantially more points than other mines. * Ghost – randomly toggles incorporeal state. Each transition time is randomly generated between 0.5 and 1 second, so it is only somewhat predictable. Collisions are only acknowledged if player collides with ghost mine while it is corporeal. Worth substantially more points than other mines. |

INHERITANCE HIERARCHY

|  |  |  |  |
| --- | --- | --- | --- |
| 1: Mine   * virtual point value * virtual movement speed * virtual force range * virtual torque range * virtual MoveForward() * abstract HandleCollide() | | | |
| 2: Normal : Mine   * Takes default values and functions | 2: Super : Mine   * Increased force and torque | | 2: E-Jump : Mine   * Normal force * No torque * E-jump++ |
| 2: Danger : Mine   * Unpredictable force * Unpredictable torque | | 3: Ghost : Super   * HandleCollide() must check if mine is ghosting, and ignore collision if it is. | |

MANAGEMENT

|  |  |  |
| --- | --- | --- |
| Invisible Loading Scene | | |
| Master Manager (MastEr)  (enduring singleton)   * All global variables * All high scores, save data, file IO * All scene management * Calls SelfDestruct() of all limited singletons if necessary | | |
| Title Manager  (limited singleton)   * All Menu UI elements |  | Game Manager  (limited singleton)   * All main game objects |
| Title Scene   * Name updates | 🡪  Player name by Global Manager | Main Scene   * Game score updates * High score checks and fireworks when beating a high score |
| 🡨  Player name by Global Manager |