

V tejto dokumentácii k nášmu zápočtovému programu popíšeme všetky gameobjecty so skriptami a ich funkciami, ktoré sú na nich pripnuté.

### ship\_01 (script ship\_movement)

- proměnné
  - objekt třídy ship\_movement uchovává několik proměnných s informacemi o hráčově postupu hrou (počet surovin na palubě, hráčovo zdraví atp.), také odkazy na objekty UI ve scéně
- vlastnosti
  - pomocí vlastností Score, counter\_coconuts a counter\_oranges manipulujeme s odpovídajícími proměnnými a překresluje odpovídající prvky UI
- metody
  - void GameOver()
    - zastaví herní čas, smaže herní soubory s mapou a stavem a zobrazí panel s výsledným skóre
  - void CheckScore()
    - porovná nové skóre s dosavadním high score. Pokud jej převyšuje, uloží se a zobrazí se hláška o dosažení nového high score.
  - void UpdateCounters()
    - přepíše aktuální stav surovin v UI
  - void TakeDamage(int damage, GameObject other)
    - odečte od zdraví utržené poškození a "odhodí" objekt třídy ship\_movement směrem od místa srážky, zároveň zatřese kamerou (pokud nezbývá žádné zdraví, vyvolá funkci GameOver())
  - void SpawnEnemy(int num)
    - vytvoří nepřátelskou loď v náhodném volném poli v okolí 5x5 ostrovů kolem hráče
  - List<Vector3> WhereCanISpawn()
    - vrátí seznam souřadnic volných bloků v okolí 5x5 bloků kolem hráče (souřadnice relativní k objektu hráče ve scéně)
  - void OnCollisionEnter(Collision collision) a void OnTriggerEnter(Collider other)
    - volají funkci TakeDamage, když hráč narazí do jiného objektu / je zasažen nepřátelskou dělovou koulí
  - void RefreshHealth()
    - překreslí ukazatel zdraví podle aktuální hodnoty proměnné health
  - IEnumerator Shoot()
    - vystřelí 3 dělové koule napravo od hráčovy lodě, upraví počet munice, zatřese kamerou a poté 4 vteřiny čeká, než povolí další výstřel
  - void Shooting()

- kontroluje, zda je možné volat funkci Shoot() (dostatek munice, zda uběhl dostatečný čas od posledního výstřelu atp.)
- *void Anchor()*
  - při stisknutí odpovídající klávesy přepíná stav kotvy (bool anchor) a odpovídající UI prvek
- *void Collect()*
  - pomocí raycastingu zjišťuje, zda kliknutí hráče směřovalo na nějaký "klikatelný" prvek; pokud ano, upraví skóre, smaže zdroj surovin atd.
- *void AddToHealth()*
  - při stisknutí odpovídající klávesy upravuje stav zdraví (int health) a pomerančů (int counter\_oranges)
- *void FindObjectInScene()*
  - do statické proměnné ship\_movement.objInScene přiřadí odkaz na objekt loď ve scéně (ten je ve scéně jediný), odkaz na objekt hráčovy loď pak mohou ostatní objekty číst odsud (odpadá potřeba pokaždé volat funkci FindObjectOfType)
- *void Start(), void FixedUpdate(), void Update()*
  - inicializují herní scénu, volají podfunkce, starají se o ovládání lodi a reakci lodi na směr větru (wind\_generator.position)

### sea (script follow\_player)

- metódy
  - *void Update()*
    - more (štvorcová plocha pod loďou) sa posúva spolu s hráčom

### wind (script wind\_generator)

- premenné
  - *Vector3 position*
    - udáva aktuálny smer vetra
- metódy
  - *IEnumerator GenerateWind()*
    - má za úlohu náhodne meniť smer vetra (Vector3 position) v scéne v určitom časovom intervale. Podľa neho sa potom mení rýchlosť plavby hráča.
  - *void Update()*
    - prehráva audio-nahrávku a mení pozíciu šípky vetra v scéne

## Main Camera (script camera\_movement)

- metódy
  - *void JumpToShip()*
    - skokom premiestni kameru nad loď
  - *void Shake()*
    - spustí korutinu *shakeTimer* so zadanou amplitudou (kamera sa zatrasie viditeľnejšie)
  - *IEnumerator shakeTimer(float amplitude)*
    - na 0.5s zvýši amplitúdu a frekvenciu kývania kamery (vyššie)
  - *void Update()*
    - Kontroluje hranicu zoomovania v scéne. Taktiež sa tu zabezpečuje pohyb kamery simulujúci vodnú hladinu / trasenie pri explózii či poškodení.

## shootbar (script shootbar)

- metódy
  - *void SetSize(float size)*
    - Nastaví veľkosť mierky nabitia kanónu v UI, teda možnosti páliť muníciu. Je volaná zo skriptu *ship\_movement*.

## healthbar (script healthbar)

- metódy
  - *void SetSize(float size)*
    - Nastaví veľkosť mierky aktuálneho zdravia. Je volaná zo skriptu *ship\_movement*.

## ball (prefab, script ball\_collision)

- metódy
  - *void OnTriggerEnter(Collider other)*
    - Detekuje kontakt munície s ostrovom alebo loďou (hráčom). Volá sa z nej korutina *DestroyBall()*, ktorá muníciu zničí až po nejakom čase, aby ešte stihol prehrať výbuch.
  - *void Update()*
    - tu sa zabezpečuje zničenie munície, ktorá spadla do mora (presiahla výškový limit)

## fortress (prefab, script enemy\_fortress)

- metódy
  - *IEnumerator Shoot()*

- Vytvorí v scéne muníciu a snaží sa zasiahnuť hráča (je zavolaná), keď sa nachádza dostatočne blízko. Toto sa zopakuje po určitom časovom intervale.
- *void Update()*
  - tu sa kontroluje, či je hráč dostatočne blízko a pevnosť môže strieľať (volať korutinu Shoot)

## menu\_handler (script menu\_handler)

- premenné
  - Pole files uchováva názvy svetov, ktoré sa aktuálne nachádzajú na zariadení.
- metódy
  - *void ArrowControl()*
    - pripnutá na arrow\_button na continue\_game\_panel a slúži na prepínanie sa medzi svetmi, ktoré si hráč na svojom zariadení vytvoril, teda mení text na choose\_world\_button
  - *void Awake()*
    - zachová game object menu\_handler pri presunutí do scény Game, aby sa v skripte generate\_islands bolo možné dostať sa k názvu sveta, ktorý si hráč vybral a tak pracovať s jeho príslušnými súbormi
  - *void BackControl()*
    - je pripnutá na back\_button na new\_game\_panel a continue\_game\_panel a má za úlohu opätovné presunutie do hlavného menu
  - *void ChooseWorldControl()*
    - figurujúca na choose\_world\_button na continue\_game\_panel prepne do scény Game, pokiaľ nejaký taký svet existuje
  - *void ContinueGameControl()*
    - je pripnutá na continue\_game\_button v hlavnom menu a zobrazuje continue\_game\_panel
  - *void ControlFiles()*
    - Má za úlohu nastaviť text na continue\_game\_panel ako meno už nejakého existujúceho sveta alebo zobrazí text „No worlds created yet“. Spúšťa sa v *Start()* a v *YesControl()*.
  - *void DeleteFiles(string name)*
    - má za úlohu vymazať svety, pre ktoré sa hráč rozhodne
  - *void DeleteWorldControl()*
    - je volaná pri stlačení tlačidla delete\_button na continue\_game\_panel a zobrazí delete\_world\_panel
  - *void Exit()*

- Slúži na ukončenie programu. Je pripnutá na exit\_button na main\_menu\_panel
- void GenerateWorld()
  - Vygeneruje nový svet a načíta scénu Game. Ak je ale meno súboru neplatné, tak zobrazí text, ktorý hovorí, že meno súboru je nesprávne. Je volaná pri stlačení tlačidla generate\_button.
- void HideInvalid()
  - Skryje text, ktorý sa zobrazí, pokiaľ hráč zadá neplatné meno súboru. Figuruje pod new\_game\_panel na name\_input\_field.
- void NewGameControl()
  - je volaná pri stlačení tlačidla new\_game\_button na main\_menu\_panel a jej úlohou je zobrazíť new\_game\_panel
- void NewWorldSlider()
  - Je volaná pri zmene hodnoty slideru a hráč si na ňom môže vybrať veľkosť svojho nového sveta. Ukáže taktiež veľkosť sveta prepočítanú na kB.
- void NoControl()
  - Deaktivuje delete\_world\_panel a continue\_game\_panel je zase viditeľný. Je pripnutá na no\_button na delete\_world\_panel pod continue\_game\_panel.
- void YesControl()
  - Deaktivuje delete\_world\_panel a continue\_game\_panel je zase viditeľný. Ďalej vymaže svet, ktorý hráč chcel a aktualizuje pole files. Je pripnutá na yes\_button na delete\_world\_panel pod continue\_game\_panel.

### pause\_panel a game\_over\_panel (script panel\_handler)

- metódy
  - void BackToMainMenuControl()
    - volaná pri stlačení na tlačidlo main\_menu\_button na game\_over\_paneli a slúži na načítanie hlavného menu po prehre
  - void MainMenuControl()
    - volaná pri stlačení na tlačidlo main\_menu\_button na pause\_paneli a ukončí hru tým, že sa uloží aktuálny pokrok hráča a následne sa hráč presunie do hlavného menu
  - void ResumeControl()
    - volaná pri stlačení tlačidla resume\_button na pause\_paneli a slúži na prerušenie hry
  - void Update()

- zisťuje, že či má hra opäť bežať alebo zostať pozastavená

## enemy (prefab, script enemy)

- metody
  - *Vector3 vectorToPlayer()*
    - vráti trojrozmerný vektor vedoucí od sebe k hráčovi
  - *bool InSight()*
    - pomocí raycastingu zjistí, zda je od něj loď hráče na dostřel (na pravé straně)
  - *void TakeDamage(GameObject other)*
    - odečte jeden bod zdraví a "odhodí" objekt třídy enemy směrem od místa srážky
  - *void Sink()*
    - posouvá objekt třídy enemy konstantní rychlostí směrem dolů (podle záporné osy y)
    - pokud je objekt dostatečně nízko (pod objektem sea), odstraní se ze scény
  - *void Emerge()*
    - posouvá objekt třídy enemy konstantní rychlostí směrem nahoru (podle kladné osy y)
  - *void OnCollisionEnter(Collision collision)* a *void OnTriggerEnter(Collider other)*
    - volají funkci TakeDamage, když objekt třídy enemy narazí do jiného / je zasažen dělovou koulí hráče
  - *void Shoot()*
    - loď vystřelí tři nepřátelské dělové koule a přehraje zvuk střelby
  - *void Update()*
    - v herní smyčce objekt kontroluje své zdraví, svou pozici a úhel, který svírá s hráčem.
    - pokud je s hráčem otočený "tím stejným směrem" (plují souběžně), snaží se dostat k jeho levému boku (tedy se k hráči natočit svou pravou stranou s kanóny),
    - pokud plují s hráčem proti sobě, snaží se dostat (z pohledu hráče) napravo.
    - loď nepřítele reaguje na vektor větru analogicky s lodí hráče.

## world\_trig (script world\_trigger)

- spravuje načítání mapy (Chunků) podle pozice hráče (vždy je načteno 9 chunků kolem hráče, při pohybu libovolným směrem se Chunky případně od/donačítají).
- proměnné
  - `Chunk[] activeChunks` - pole 9 aktivních Chunků

- `int arrayStart` - index "horního levého" Chunku v poli `activeChunks`, aby nebylo potřeba při každé změně přesouvat všechny prvky v poli
- metody
  - `void SpawnPlace(int x, int y)`
    - načte chunky kolem hráče (`SpawnPlace` se volá z `generate_islands.Start()`, tedy po už načtení stavu/vytvoření světa), přemístí hráče a kameru na danou pozici
  - `void OnTriggerExit(Collider collider)`
    - když objekt `world_trig` (ve scéně) zaznamená "kolizi" s objektem hráče (tedy když se hráč dostatečně vzdálí od středu prostředního aktivního Chunku), zjistí, kterým směrem se hráč pohybuje; tímto směrem vygeneruje nové Chunky (na Chunky na protější straně zavolá jejich metodu `RemoveIslands()`).

### **class Chunk (neodpovídá žádnému objektu ve scéně)**

- spravuje bloky herního světa ve scéně (1 chunk = 16x16 ostrovů), manipuluje s ostrovy ve scéně
- proměnné
  - `int pos_x, pos_y` - pozice Chunku ve scéně
  - `List<GameObject> existingIslands` - seznam podřízených ostrovů (objektů ve scéně)
  - vedle toho třída `Chunk` obsahuje několik statických proměnných s informacemi o typech ostrovů, jejich možných rotacích + objektech zájmu (stromy + pevnosti).
- metody
  - `Chunk(int pos_x, int pos_y)` - konstruktor,
    - přiřazuje do proměnných informace o pozici Chunku ve scéně
  - `void InstinScene()`
    - `Chunk` čte "svůj" (podle své pozice ve scéně) 16x16 výsek mapy (`WorldLoader.world_map`) a podle přečtených dat vkládá do scény dané ostrovy (+ případné objekty zájmu).
    - zároveň nové objekty ve scéně přidává do seznamu `existingIslands` (i objekty zájmu!)
  - `void RemoveIslands()`
    - `Chunk` projde seznam `existingIslands` a na každý objekt v něm zavolá funkci `Destroy()` - vymaže své objekty ze scény.
  - `void printChunk()`
    - vytiskne příslušnou část mapy (`Chunk`) do `Debug.Log()`

- *int TypeOfIsland(int x, int y)*
  - zjistí TVAR (ne typ) ostrovu na dané pozici v Chunku (tzn. kolik ostrovů s ním sousedí a kde jsou => číslo 0-15 odpovídající tvaru a rotaci ostrovu (pole `Chunk.models[]`) tak, aby se na své sousedy "napojil").

### **class WorldLoader (neodpovídá žádnému objektu ve scéně)**

- stará se o načítání herního světa ze souborů a o manipulaci s mapou světa.
  - (všechny proměnné i metody jsou statické)
  - proměnné
    - `byte[][] world_map` - odpovídá aktivní "mapě ostrovů" (matici typů ostrovů)
      - 0x00 = moře
      - 0x01 = prázdný ostrov
      - 0x02 = ostrov s pomerančovníkem
      - 0x03 = ostrov s palmou
      - 0x04 = ostrov s pevností
  - `int world_size` - odpovídá jednomu rozměru `world_map`
  - `string activeMapFilename` - jméno aktuálního souboru s mapou (bez přípony)
  - `int spawnX, int spawnY` - počáteční/uložená pozice hráče (pokud se otevírá již hraná mapa, přečte se z `.state` souboru)
  - metody
    - *void SaveState()*
      - volá funkci `SaveState(string filename)` s jménem aktivní mapy
    - *void SaveState(string filename)*
      - zapisuje do souboru `filename.state` aktivní stav hry (pozici hráče ve světě, počet surovin, zdraví, skóre)
      - soubor `.state` je textový soubor, informace jsou odděleně na jednotlivých řádcích
    - *int mod(int x, int m)*
      - provádí operaci modulo (i se zápornými čísly)
    - *int[2] CoordsToMatrix(Vector3 coords)*
      - z pozice objektu ve scéně (`coords`) vypočítá index "buňky" ve `world_map`, které tato pozice odpovídá
    - *byte ReadFromMap(int y, int x)*
      - vrátí "typ" ostrovu, který leží na daném indexu (modulo `world_size`) ve `world_map`
    - *void LoadState(string filename)*



- přečte .state soubor se zadaným jménem a nastaví podle něj proměnné objektů v aktuální scéně
- *void CreateMapMatrix(int size)*
  - inicializuje pole world\_map podle zadané velikosti
- *void WriteMap(string filePath)*
  - uloží aktuální mapu (world\_map) do (binárního) souboru filePath.world.
  - world\_map se čte po řádcích, v řadě se ukládá do souboru. Prvních 16 bytů je hlavička souboru, její první 4 byty jsou velikost mapy (world\_size)
- *void ReadMap(string filePath)*
  - načte soubor filePath.world do aktuální mapy (world\_map)
  - nejdříve přečte hlavičku, podle ní pak inicializuje world\_map dané velikosti a naplní její data ze souboru
- *void PerlinGenerate()*
  - naplní mapu (world\_map) "ostrovy" s pomocí Perlinova šumu (pokaždé s náhodným offsetem, aby bylo rozložení ostrovů různé v různých mapách)
  - s pomocí funkce RandomPOIDistribution() pak na nenulové bloky (ostrovy) přidá "points of interest" - pevnosti, stromy atd.
- *byte RandomPOIDistribution()*
  - s neuniformním rozdělením vygeneruje číslo od 1-4 (odpovídá typům ostrovů)
- *void LoadMap(string filePath, int size)*
  - načte zadanou mapu ze souboru filePath.world, popřípadě tento soubor vytvoří, pokud mapa neexistuje (zde využije parametr size)