UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Grafică, Multimedia și Realitate Virtuală

PROIECT DE DIPLOMA

ÎNTRUMĂTOR ȘTINȚIFIC
Prof. Dr. Ing. Moldoveanu Alin

ABSOLVENT
Soare Robert Daniel

Bucuresti, 2023

UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Grafică, Multimedia și Realitate Virtuală

Aprobat Decan ` Prof. Dr. Ing. Mihnea Alexandru Moisescu

PROIECT DE DIPLOMA

Design extensibil pentru Tower Defense

ÎNTRUMĂTOR ȘTINȚIFIC
Prof. Dr. Ing. Moldoveanu Alin

ABSOLVENT
Soare Robert Daniel

Bucuresti, 2023

Cuprins

Introducere

Genul „tower defense” (sau „apărare prin turnuri”) este un gen de joc video în care jucătorul are rolul de a construi și de a upgrada turnuri defensive pentru a împiedica invazia inamicilor și a proteja o anumită zonă sau resurse. Jucătorul trebuie să plaseze strategic turnuri cu diferite abilități și funcții, cum ar fi turnuri de tragere, turnuri de aruncare a proiectilelor sau turnuri magice, pentru a opri inamicii să ajungă la punctele cheie ale hărții sau să distrugă baza jucătorului. Pe măsură ce jocul avansează, inamicii devin tot mai puternici, iar jucătorul trebuie să-și îmbunătățească strategiile de apărare și să facă alegeri strategice pentru a reuși să reziste valurilor de inamici.

De-a lungul anilor, genul de joc Tower Defense a evoluat și a rămas unul dintre cele mai populare genuri de jocuri în rândul jucătorilor din întreaga lume. Cu toate acestea, odată cu creșterea continuă a pieței jocurilor video, există nevoia de a inova și de a oferi jucătorilor o experiență de joc nouă și interesantă. Iar această experiență poate varia de la o valoarea educativă [1] la una pur distractivă.

Acest gen de joc este tot mai întâlnit pe platformele de jocuri mobile, precum Google Play și App Store. Punctele forte care îl fac să fie atât de popular sunt:

- Sesiuni de joc scurte care pot fi jucate oricând și oriunde.
- Nu necesită atenție continuă, jucătorul poate să se concentreze pe alte activități în timp ce jocul rulează în fundal.
- O rundă de joc poate fi câștigată prin mai multe moduri, astfel oferă o experiență de joc variată.
- Mecanică joc de simplu de învățat.
- Joc de strategie care îi oferă jucătorului satisfacția de a reuși să reziste valurilor de inamici prin prisma deciziilor strategice pe care le ia.

Un punct slab al acestui gen de joc este găsirea unui echilibru în relația dintre turnurile de apărare. In marea majoritate a jocurilor de aceste gen care au avut succes, turnurile de apărare sunt independente și nu au nevoie de ajutorul unui alt turn pentru a funcționa. Acest lucru simplifică mecanica jocului, însă poate duce la o experiență de joc monotonă. Implementarea unui sistem interdependent aduce un grad de complexitate ridicat atât pentru jucător cât și pentru dezvoltator. Cea mai mare problemă fiind design hărții de joc care trebuie să fie concepută astfel încât să pună în evidență interdependența turnurilor de apărare.

În această lucrare, ne propunem să cercetăm un model semi-interdependent de colaborare pentru turnurile de apărare. Acest model va folosii un sistem de comunicare bazat pe mesaje pentru a comunica între turnuri. Iar aceste mesaje vor fi purtate de către inamici cu denumirea de **jetoane de acțiune**. Astfel, turnurile vor avea acțiuni care vor fi declanșate de aceste jetoane, iar sarcina jucătorului este să se asigure că aceste jetoane ajung la turnurile potrivite pentru a declanșa acțiunea dorită.

Așadar, această lucrare va descrie un set de specificații pentru un joc de tip Tower Defense care va implementa un sistemul de colaborare între turnurile de apărare care va urma să fie descris. Pe langă design-ul jocului, vom prezenta și cum această idee poate fi structurată și implementată într-un motor de joc.

1. Genul de joc Tower Defense

Tower Defense este un gen de joc de strategie în care jucătorii trebuie să-și construiască și să-și îmbunătățească turnuri defensive pentru a împiedica inamicii să ajungă la obiectiv.

Jocurile de Tower Defense se desfășoară de obicei pe o hartă cu un traseu prestabilit pe care inamicii încearcă să avanseze, în timp ce jucătorii plasează și își upgradează turnurile defensive pentru a-i opri. Fiecare turn are caracteristici unice, cum ar fi raza de acțiune și puterea de foc, și trebuie plasat strategic pentru a maximiza efectivitatea sa. Jucătorii trebuie să ia decizii strategice importante în timpul jocului, cum ar fi ce tipuri de turnuri să construiască, când să le construiască și cum să le upgradeze, pentru a se asigura că pot gestiona cu succes amenințările inamice.

Genul de joc Tower Defense își are rădăcinile într-un mod popular de joc numit „Maze Games" care a început să fie jucat în anii 1980. În aceste jocuri, jucătorii trebuiau să-și construiască un traseu labirintic pentru a împiedica inamicii să ajungă la destinație. Aceste jocuri au fost inspirația pentru dezvoltarea jocurilor de Tower Defense, care au început să apară în anii 1990, în special în Japonia. Prin intermediul modurilor create pentru jocul *Warcraft 3: The Frozon Throne*, popularitatea acestui gen a fost sporită și acestea au deservit ca sursă de inspirație pentru viitoarle titluri care urmau să apară [2].

Punctele puternice ale acestui gen de joc sunt:

- Gameplay-ul relativ simplu și ușor de înțeles, ceea ce le face accesibile pentru toți jucătorii, indiferent de nivelul lor de experiență.
- Sesiuni scurte de joc împărțite în niveluri relativ scurte, ceea ce le face perfecte pentru a fi jucate în timpul pauzelor de la muncă, în transportul public sau în orice moment liber.
- Implică planificarea, gestionarea resurselor și luarea deciziilor strategice pentru a proteja o anumită zonă de atacurile inamicilor.

Alte motivații pot fi [3]:

- Creativitatea - Fiecare jucător își decide propriul mod a aborda jocul.
- Complexitatea - Dorința de a depăși obstacolele impuse de mecanisme de joc complexe.
- Bucuria procesului - Procesul de rezolvarea a jocului este satisfăcător în sine.

În următoarea listă avem exemple de jocuri populare și emblematice ale acestui gen de joc care au aparut de-a lung timpului:

- Plants vs. Zombies - În acest joc, jucătorii trebuie să își planteze flori și alte plante pentru a împiedica zombii să ajungă la casa lor. Plantele au rol de turn de apărare, iar fiecare are abilități unice. Jucătorii trebuie să le plaseze strategic pentru a opri zombii înainte ca aceștia să ajungă la locuințele lor. Jocul oferă o varietate de niveluri și moduri de joc. Acest a fost dezvoltat de către PopCap Games și a fost lansat în anul 2009.
- Kingdom Rush - Acest joc are loc într-o lume de basm, unde jucătorii trebuie să își construiască turnuri defensive și să își organizeze trupele pentru a învinge hoardele de creaturi mitice. Jocul oferă o gamă largă de turnuri și trupe, fiecare cu abilități și caracteristici unice. De asemenea, aceste este unul dintre cele mai emblematice jocuri ale genului de joc Tower Defense. Jocul a fost lansat în anul 2011 și a fost dezvoltat de către Ironhide Game Studio. Au mai fost lansate și alte versiuni ale jocului, cum ar

fi Kingdom Rush: Frontiers, Kingdom Rush: Origins și Kingdom Rush: Vengeance care s-au bucurat de un succes imens.

- Bloons TD 6 - Jocul este centrat pe o tematica simple in care trebuie sa apare baza de baloane, jucătorii trebuie să își plaseze turnurile defensive (care au forma unor maimute) și să își upgradeze abilitățile pentru a împiedica baloanele să ajungă la final. Jocul oferă o mulțime de turnuri și abilități, precum și niveluri diferite cu diverse provocări. Jocul a avut parte de mai multe versiuni de-a lungul anilor. Versiunea 6 este ultima lansată și care este îmbunătățită în continuare. Acest a fost dezvoltat de către Ninja Kiwi și a fost lansat în anul 2018.
- Orcs Must Die! - O combinație interesantă între jocuri de strategie și jocuri de acțiune. Jucatorul are posibilitatea să participe în mod activ la oprirea valului de inamici prin intermediul unui personaj. Jocul oferă o gamă largă de arme și abilități pentru personaje, iar jucătorii trebuie să le folosească strategic pentru a supraviețui nivelelor și a învinge inamicii. Harta de joc se aseamană unui tunel, iar partea creativă constă în faptul cu turnurile de aparare (care apar sub forma unor capcane) pot fi poziționate pe podea, pereți și tavan. Jocul a fost lansat în anul 2011 și a fost dezvoltat de către Robot Entertainment.
- Dungeon Defenders - Acest joc combină elemente de joc de rol și Tower Defense, jucătorii trebuie să-și construiască turnurile defensive și să-și antreneze eroii pentru a învinge inamicii. Jocul oferă o gamă largă de personaje cu abilități unice, iar jucătorii trebuie să le upgradeze și să le echipeze înainte de a se aventura în nivelele următoare. Acesta prezintă o combinație interesantă între strategie și joc de rol. Jocul a fost lansat în anul 2011 și a fost dezvoltat de către Trendy Entertainment.
- Factorio - Acest joc este un Tower Defense în care jucătorii trebuie să își construiască o bază și să o apere de atacurile inamicilor. Jocul oferă o gamă largă de structuri defensive, precum și posibilitatea de a le upgrade. Jucătorii trebuie să colecteze resurse și să le folosească pentru a construi structuri defensive. Principală trăsătura a jocului este complexitatea lanțului de aprovizionare pentru clădirile de producție a resurselor complexe. Jocul a fost lansat în anul 2016 și a fost dezvoltat de către Wube Software.

2. Design-ul unui joc de Tower Defense

Design unui joc de Tower Defense este relativ simplu. În general, jocurile de Tower Defense au următoarele elemente:

- O bază care trebuie apărată de atacurile inamicilor.
- O hartă cu un traseu pe care inamicii încearcă să avanseze.
- Inamici care atacă baza jucătorului.
- Turnuri defensive care trebuie plasate strategic pentru a opri inamicii înainte ca aceștia să ajungă la baza jucătorului.
- Resurse care trebuie colectate pentru a construi turnurile defensive.

O reprezentare simplificată poate fi observată în Figura 1.

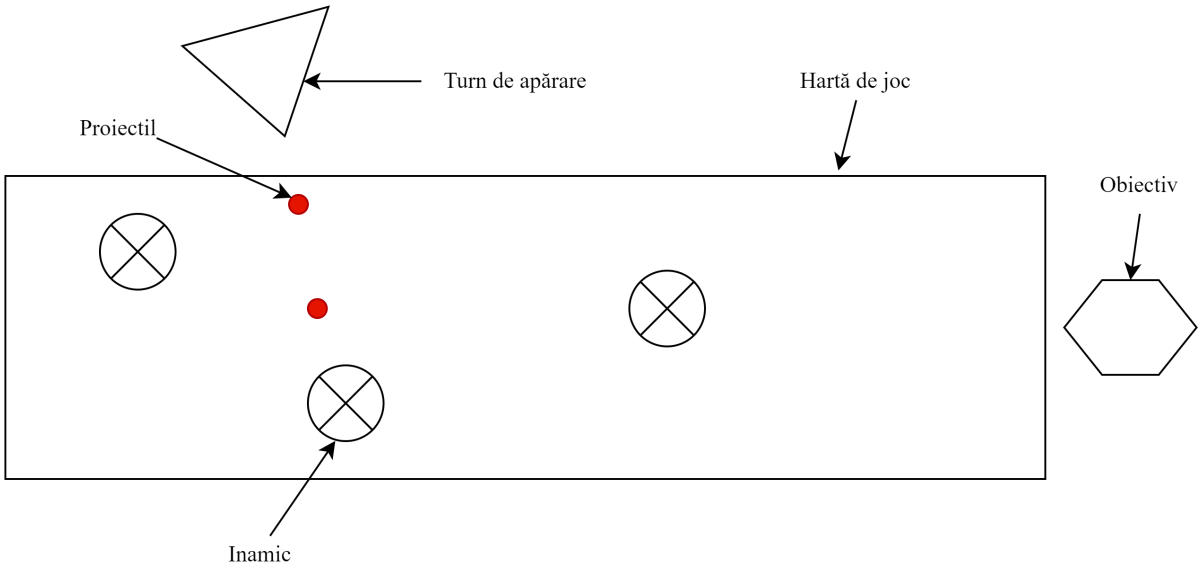

Figura 1: Schiță pentru un element vizual al unui îmbunătățiri din magazin.

Alte aspecte care pot fi luate în considerare în design-ul unui joc de Tower Defense sunt [3]:

- Pagubele cauzate de turnuri asupra inamicilor - fiecare inamic are un anumit număr de puncte de viață, iar turnurile au un anumit număr de puncte de atac. Atunci când un inamic este atacat de un turn, acesta pierde puncte de viață. Atunci când punctele de viață ale unui inamic ajung la 0, acesta este eliminat din joc. Trebuie să avem în vedere acest lucru atunci când proiectăm turnurile de apărare, deoarece acestea trebuie să fie suficient de capabile să elimine inamicii înainte ca aceștia să ajungă la obiectiv.
- Modul de tințire - uneori dorim ca tunurile să atace anumiți inamicii înaintea altora. Acest lucru poate fi realizat prin intermediul unui sistem de prioritizare a inamicilor. De exemplu, un turn poate fi configurat să atace întotdeauna inamicul cel mai apropiat de baza jucătorului, sau poate fi configurat să atace întotdeauna inamicul cu cele mai multe puncte de viață.
- Optimizarea cheltuielilor de resurse - fiecare rundă de joc oferă o anumită cantitate de resurse. Costul turnurilor trebuie să țină cont de această voloare întrucât jucătorii trebuie să poată construi turnuri în fiecare rundă. Prea multe resurse pot duce la un joc prea ușor, iar prea puține resurse pot duce la un joc prea dificil.
- Particularități are hărții de joc - unele hărți de joc pot avea particularități care pot influența modul în care jucătorii își construiesc turnurile de apărare. De exemplu, o hartă de joc poate avea un traseu care

se împarte în două, iar jucătorii trebuie să își construiască turnurile de apărare în așa fel încât să poată apăra ambele trasee.
- Efecte de control a mulțimii - unele turnuri pot avea efecte de control a mulțimii asupra inamicilor. De exemplu, un turn poate încetini inamicii, sau poate îngheța inamicii pentru o perioadă de timp. Aceste efecte pot fi foarte utile în anumite situații, de exemplu, atunci când un inamic este foarte aproape de obiectiv.
- Armura inamiciilor - un sistem prin care inamicii pot diminua efectele produse de la anumite tipuri de atacuri. De exemplu, un inamic poate fi rezistent la atacurile de foc, dar poate fi vulnerabil la atacurile de gheață. Acest sistem este de multe ori introdus pentru a încuraja jucătorii să își construiască turnuri defensive de diferite tipuri.

Design-ul jocurilor de Tower Defense a evoluat semnificativ în ultimii ani. Iată câteva exemple de evoluție a design-ului pentru jocurile de Tower Defense:

- Varietate în tipurile de structuri defensive. Creativitatea dezvoltatorilor a fost foarte inovatoare pentru acest aspect. Multe jocuri asemănand turnurile de apărare cu alte structuri, cum ar fi: capcane, arme, aparate sau chiar personaje. Acest lucru a oferit o flexibilitate în dezvoltarea jocurile hibride care se îmbină cu alte genuri de jocuri.
- Inamici pot fi rezistenți la anumite tipuri de atacuri sau pot avea abilități de evitare a atacurilor din partea structurilor defensive.
- Designul hărții de joc si-a păstrat structura de bază, schimbările au fost mai pronunțate în ceea ce privește tematica și complexitatea treseului pentru inamici. Unele hărță având trăsături unice care influențează modul în care jucătorii își plasează structurile defensive.
- Economia de joc a fost extinsă, introducerea mai multor tipuri de resurse și a unor mecanisme de colectare mai complexe au avut un impact pozitiv în cea privește partea strategică de gestionare a resurselor. Mulți dezvoltatori folosindu-se de acesta pentru a integra elementele din jocurile de tip „resource management”.

2.1. Modele hibride de jocuri

Jocurile de Tower Defense au evoluat și au început să se îmbine cu alte genuri de jocuri, ceea ce a dus la apariția unor noi genuri de jocuri. Iată câteva exemple de modele hibride:

- Joc de dezvoltarea a unei base (*base building*). Acesta de bazează pe colectarea de resurse și crearea de unui lanț de aprovizionare pentru clădirile de producție. Este o situație destul de comună ca aceste tipuri să jocuri să includă o parte de Tower Defense. Accentul se pune pe partea economică, obiectivul jucătorului fiind deplocarea unor clădri speciale care au nevoie de resurse complexe pentru a fi construite. Sistemul de jetoane de acțiunea se folosește aceiași idee, număi că în loc să creăm un lanț de aprovizionare, jucătorii trebuie să creeze un lanț de acțiuni care să fie executate într-o anumită ordine pentru a obține un rezultat dorit.
- Joc de rol și acțiune (*action role-playing game*). Acestea aduc in prim-plan partea de poveste și acțiune continuă pentru a crea o experiență de joc mai interesantă. Ele se concentrează pe crearea unui univers care să-l captiveze pe jucător. Jocurile de acest tip oferă o varietate de eroi cu abilități și caracteristici unice care se îmbină cu tematica elementara a genului Tower Defense.
- Puzzle. Acest gen se pliază foarte ușor pe genul Tower Defense, iar introducerea de mici schimbări în mecanica de joc poate duce la crearea unui joc de acest tip. Acestea concentrează pe crearea unui puzzle care să fie rezolvat de către jucător. Acest puzzle poate consta în găsirea unei anumite combinații

de structuri defensive care să oprească valul inamic în anumite condiții. Unele jocuri, introduc acest concept sub form de „challenge mode” care poate fi jucat de către jucători după ce au terminat jocul. Acesta consta în adaugarea de noi constrângeri pentru jucător, cum ar fi: limitarea numărului de turnuri defensive, limitarea numărului de resurse, creșterea numărului de inamici, etc.

- Joc de strategie în timp real (*real-time strategy*). Acest gen de jocuri se concentrează pe crearea unei strategii de joc care să fie executată în timp real. O mare partea din aceastea au o parte competitivă dezvoltată unde fie care greseală poate fi exploatată. Mai toate jocurile de acest timp conțin turnuri de apărare care rolul de a încetinii atacul oponentului.

2.2. O nouă metodă de extindere a mecanicii de joc

Având în vedere toate variantele prin care a evoluat genul Tower Defense, încă există arii care nu au fost explorate. Unul dintre acestea este crearea unui sistem colaborativ între turnurile de apărare.

În mare majoritate a jocurilor create, turnurile de apărare funcționează independent, fiecare având propriile sale abilități și caracteristici. Unele jocuri au introdus mici schimbări, prin care turnurile de apărare pot fi îmbunătățite, iar unele din îmbunătățiri fiind în a oferi turnurilor vecine mici bonusuri.

Pentru a dezvolta această arie, explorăm următoarea idee de mecanică de joc: Fiecare turn de apărare poate crea și consuma un **jeton de acțiune** de pe inamicul din raza de apărare a turnului. Un jeton de acțiune reprezintă o acțiune care poate fi efectuată de unele turnuri de apărare și acesta este purtat de către inamicii. Un turn de apărare poate crea un jeton de acțiune care poate fi consumat de un alt turn de apărare prin intermediul inamicilor.

Practic, dezvoltăm un sistem de comunicare între turnuri, care reacționează diferit la prezența unui inamic în funcție de tipul și numărul de jetoane de acțiune pe care le dețin. Fiecare turn are capacitatea de a genera un jeton de acțiune atunci când interceptează un inamic în raza sa, iar acest jeton poate fi ulterior utilizat de un alt turn pentru a efectua o acțiune specifică. De exemplu, un turn poate produce un jeton de acțiune care declanșează un atac special.

Harta joc are un rol crucial în design, ea este cea care de cele mai multe ori oferă unicitate unui sesiuni de joc. În multe jocuri clasice, dimensiunea mărită a hărții de joc nu oferă prea multe oportunități deoarece de multe ori jucătorul aplică aceiași combinație turnuri, fapt care crează redundanță. Trecând către acest sistem colaborativ, putem sesiza un nou beneficiu, și anume: hărțile de dimensiune mai mare ne oferă mai mult spațiu pentru a crea mai multe turnuri defensive, fapt care ne permite să creăm mai multe interacțiuni între acestea.

Următoarele întrebări apar în urma acestei idei de mecanică de joc:

- Cum putem echilibra macanica de joc? Dacă un jeton este mult mai bun decât celalte, cum le putem face pe celelalte mai atractive?
- Cum putem crea o varietate de jetoane de acțiune care să fie interesante pentru jucător fără a crea redundanță?
- Care este dimensiunea hărții de joc are să poată susține această mecanică de joc care necesită un număr mare de turnuri de apărare?
- Care este numărul de jetoane optime pentru fiecare tip?
- Cum aratăm acest sistem în interfața de utilizator astfel încât să fie ușor de înțeles?
- Cum structurăm turnurile de apărare astfel încât să evidențiem acest sistem de jetoane de acțiune?

2.3. Compunerea jetoanelor de acțiune

Flexibilitea jetoanelor de acțiune vine din faptul că acestea pot fi compuse din mai multe tipuri de jetoane. Presupunem că avem următoarele tipuri de jetoane de bază:

- Jeton de bonus atac: inamciii primesc daune bonus de la proiectilele turnurilor.
- Jeton de încetinire: inamicii au viteză de mișcare redusă.
- Jeton de explozie: proiectilele turnurilor au un efect de explozie asupra inamicului și vecinilor săi.

Rangul unui jeton reprezintă numărul de jetoane de același tip pe care un inamic le deține, fiecare jeton are prestabilit un rang maxim. Un jeton de rang mai mare are un efect mai pronunțat decât unul de rang mai mic. Jetoanele pot fi combinate pentru a obține noi tipuri. De exemplu, putem avea următoarele tipuri de jetoane compuse:

- Jeton de înghețare: inamicii vor fi înghețați (viteză de mișcare 0) atăta timp căt jetonul este activ. Format dintr-un jeton de încetinire de rang 2.
- Jeton de daune de-a lungul timpului: inamicii care dețin acest jeton vor primi daune atâta timp cât jetonul este activ. Format dintr-un jeton de bonus atac de rang 3.
- Jeton de explozie la eliminare: inamicii care dețin acest jeton vor crea o explozie când sunt eliminați. Format dintr-un jeton de explozie de rang 2.
- Jeton de explozie la încetinire: inamicii care dețin acest jeton vor crea o explozie când sunt încetiniți. Format dintr-un jeton de explozie de rang 2 și un jeton de încetinire de rang 2.
- Jeton de explozie pulsantă: inamicii care dețin acest jeton vor crea o explozie în jurul lor la fiecare 3 secunde. Format dintr-un jeton de explozie de rang 2 și un jeton de bonus atack de rang 3.

Putem observa marele avantaj al acestui sistem de jetoane de acțiune, și anume: flexibilitatea. Acest sistem ne permite să creăm o varietate de jetoane de acțiune, iar acestea pot fi combinate pentru a crea noi tipuri de jetoane.

Așadar, un joc care urmează acest design poate fi extins foarte ușor fără schimbări majore în mecanica de joc. Această flexibilitate poate fi observată în jocurile de tip cărți de joc precum: *Hearthstone* sau *Magic: The Gathering*. Aceste jocuri au o mecanică de bază simplă, însă prin intermediul extensiilor, acestea pot fi extinse cu noi tipuri de cărți care aduc varietate.

Însă, acest sistem de jetoane de acțiune nu este fără dezavantaje. Următoarele probleme pot apărea:

- Conflicte la compunere: presupunem ca inamicul se află în raza a doua turnuri care folosesc același tip de jeton (Figura 2). Se pune problema care dintre cele două va avea prioritate la procesarea jetonului.
- Redundanță: este destul de dificil să avem o multitudine de jetoane fără să existe unele care fac aproximativ același lucru preacum altele.
- Logică complexă: toate aceste interacțiunii au nevoie să fie procesate iar construirea unui sistem care să facă acest lucru poate fi destul de dificilă având în vedere că efectele trebuie să fie aplicate într-o anumită ordine.

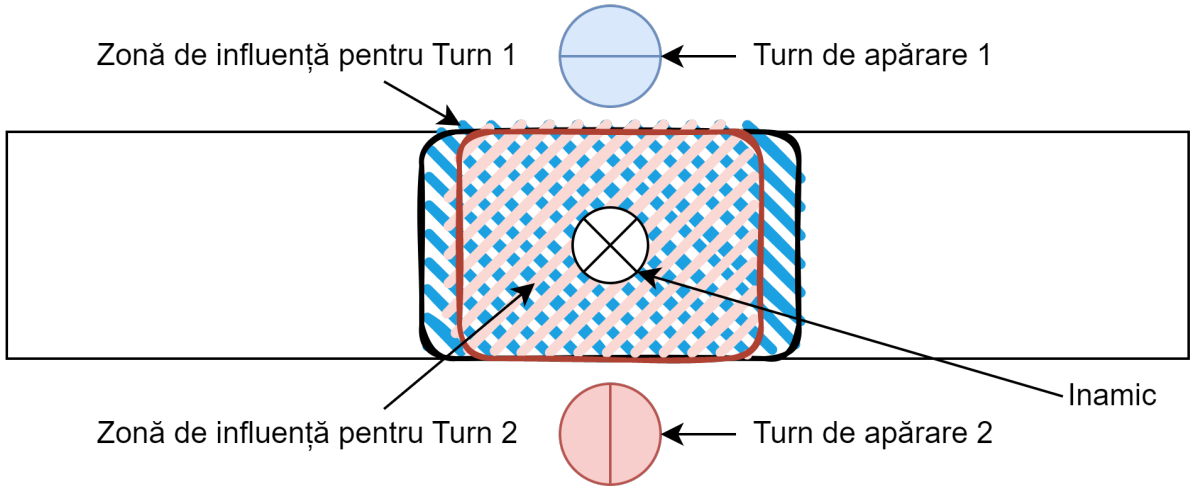

Figura 2: Conflict la compunere

2.4. Tipurile de turnuri

Pentru sistemul de jetoane de acțiune trebuie să avem o gamă largă de turnuri care să fie capabile să creeze și să consume jetoanele de acțiune. Vom împărți turnurile în două categorii: *active* și *pasive*.

Proiectilele pot fi create doar de către turnurile active și acestea reprezintă principalul mod a elimina inamicii. Dar turnurile active nu pot crea jetoane de acțiune, ele pot doar consuma jetoanele de acțiune pentru a-și îmbunătăți atacul. Turnurile pasive nu pot crea proiectile, însă acestea pot crea jetoane de acțiune care pot fi consumate de către turnurile active sau de alte turnuri pasive.

Această diviziune între turnurile active și turnurile pasive aduce o dimensiune suplimentară strategiei jocului de *Tower Defense*. Jucătorul trebuie să decidă cum să își aloce resursele și să plaseze turnurile în mod strategic pentru a crea un echilibru între puterea ofensivă și capacitatea de a genera jetoane de acțiune. Relație de interdependență între turnuri este un aspect important în design-ul jocului, el fiind forma de coloborare care este create de acest sistem de jetoane.

În acest sens, putem avea următoarele tipuri de turnuri pasive din Tabelul 1:

- Turn pasiv de atac bonus: crează un jeton de bonus atac pentru fiecare inamic din raza sa de acțiune.
- Turn pasiv de încetinire: crează un jeton de încetinire pentru fiecare inamic din raza sa de acțiune.
- Turn pasiv de înghețare: crează un jeton de înghețare pentru fiecare inamic din raza sa de acțiune care are un jeton de încetinire de rang 2.
- Turn pasiv de explozie pulsantă: crează un jeton de explozie pulsantă pentru fiecare inamic din raza sa de acțiune care are un jeton de explozie de rang 2.

Turnurile active se aseamănă cu turnurile clasice de apărare, acestea având rolul de a elimina inamicii. Dacă turnurile pasive variază prin tipul de jeton pe care îl produc, turnurile active se vor diferenția prin modelul de proiectil create și rata de atac. Iată câteva exemple de turnuri active:

- Turn activ cu proiectil simplu: acesta crează un proiectil simplu care aplică pagube primului inamic cu care intră în contact. Acesta atacă la o rată medie și aplică pagube medii.
- Turn activ de tip mortar: acesta crează un proiectil care explodează la impact și aplică pagube tuturor inamicilor din raza de acțiune a exploziei. Proiectilul explodează când ajunge la destinație, acesta ignorând inamicii din cale.
- Turn activ cu atac rapid: acesta crează un proiectil simplu care aplică pagube primului inamic cu care intră în contact. Acesta are o rată de atac ridicată, dar care provoaacă pagube mici.

- Turn activ cu proiectil inteligent: acesta crează un proiectil simplu care urmărește inamicul cel mai apropiat din raza sa de acțiune.
- Turn activ cu proiectil penetrant: Acest tip de turn creează proiectile care pot străpunge inamicii și pot atinge și dauna mai mulți inamici în linie. Proiectilele penetranți sunt deosebit de eficiente împotriva inamicilor cu armură sau a grupurilor de inamici care avansează într-o linie.

Având în vedere exemple de mai sus, putem observa diferite cum fiecare turn îl completează pe celălalt:

- Pentru ca turnul activ să fie eficient, ar avea nevoie de niște turnuri pasive care să încetinească inamicii.
- Turnul de tip mortar ar fi mai bun dacă inamicii ar fi mult mai încetiniți astfel încât aceștia să fie mai grupați astfel încât explozia să fie mai eficientă.
- Turnul cu proiectil inteligent nu ar avea probleme cu țintirea inamicilor așă că ar beneficia mai mult dacă inamicii ar avea un jeton de explozie pulsantă care să le reducă viața cât mai repede.

Denumire turn pasiv	Condiție (jetoane de acțiune)	Jeton generat
Atac bonus	-	Bonus atac (rang 1)
Încetinire	-	Încetinire (rang 1)
Înghețare	Încetinire (rang 2)	Înghețare (rang 1)
Explozie	-	Explozie (rang 2)
Explozie pulsantă	Explozie (rang 2)	Explozie pulsantă (rang 1)

Tabelul 1: Exemple de turnuri pasive

2.5. Valul de inamici

Scopul unui inamic este să ajungă la obiectivul pe care jucătorul încearcă să-l protejeze. Pentru a ajunge la destinație, acesta trebuie să supraviețuiască atacurilor turnurilor de apărare. Inamicii au o viață și o viteză de mișcare. Viața reprezintă numărul de pagube pe care un inamic le poate suporta înainte de a fi eliminat. Viteza de mișcare reprezintă cât de repede se deplasează inamicul pe traseu.

Ca un inamic să ajungă la obiectiv, viața lui trebuie să fie mai mare decât pagubele pe care le pot produce turnurile de apărare de-a lungul traseului. Sau, poate fi mai mică, însă viteza de mișcare trebuie să fie mai mare decât viteza proiectilelor. Sunt multe moduri în care putem varia caracteristicile sale.

Proiectarea corectă a inamicilor este un aspect important în design-ul jocului. Inamicii trebuie să fie echilibrați astfel încât să ofere o provocare jucătorului și să fie în armonie cu sistemul de jetoane. Pentru inamicii nu avem un anumit tip ci un set de recomandări de design de care trebuie ținut cont:

- Un inamic nu poate fi eliminat doar prin intermediul turnurilor active – ne dorim să existe o coloborare între turnuri, așadar turnurile pasive trebuie să aibă și ele o contribuție.
- Caracteristicile inamicului (precum: viață, viteză de mișcare) trebuie să fie în concordanță cu evoluția jocului. Inamicii devin mai puternici pe măsură ce jocul avansează.
- Generarea valului de inamici trebuie să fie consistent. În loc să generăm aleatoriu pozițiile de start al inamiciilor, putem folosi un algoritm de generare care să producă un traseu de la punctul de start la punctul final. Acest lucru ne permite să controlăm mai bine dificultatea jocului.

Unele jocuri, introduc noi mecanici de joc pentru inamicii, precum:

- Armură: atacurile de un anumit tip au un efect redus asupra inamicilor care au armură (exemplu: -50% pagube primite de la proiectil). Aceasta, poate să introducă la rândul său noțiune de *tipuri de atac* (exemplu: atac de foc, atac magic) unde fiecare tip de armură acționează diferit.
- Regenerare: inamicii își pot regenera viața în timpul jocului.
- Abilități speciale: inamicii care oferă un bonus altor inamici din jurul lor (exemplu: +50% viață pentru inamicii din jurul său) sau care produc o acțiune care le oferă avantaj (exemplu: crearea de noi inamicii de-a lungul traseului).

Sistemul de jetoane nu prezintă niciun impediment în implementarea acestor mecanici de joc. Chiar putem crea jetoane de acțiune care să contracareze inamicii care prezintă aceste mecanici de joc. De exemplu, putem avea un jeton de acțiune care să reducă armura inamicilor sau un jeton de acțiune care să reducă regenerarea inamicilor.

Fiecare rundă de joc are un număr de inamicii care compun valul de inamicii (Tabelul 2). Acest număr crește pe măsură ce jocul avansează. De exemplu, în primele 10 runde, valul de inamicii este format din 10 inamicii, în următoarele 10 runde, valul de inamicii este format din 25 inamicii.

Interval runde	Număr inamici per rundă
1-10	10
11-20	25
21-30	40
31-40	60

Tabelul 2: Numărul de inamicii care compun un val per rundă

2.6. Economia de joc

Într-un joc de tip *Tower Defense*, turnurile de apărare sunt construite și îmbunătățite prin intermediul resurselor. Se utilizează diverse tipuri de resurse pentru a conferi jucătorului un control strategic asupra *gameplay*-ului. Un astfel de element esențial este moneda, reprezentând o unitate de valoare în cadrul jocului, utilizată pentru achiziționarea și îmbunătățirea turnurilor de apărare. Aceasta poate fi obținută prin intermediul unor mecanici specifice, cum ar fi distrugerea inamicilor sau îndeplinirea cu succes a anumitor provocări.

Acest sistem joacă un rol important în partea strategică a jocului. Gestionarea atentă a resurselor reprezintă aspecte cheie în dezvoltarea unui joc de *Tower Defense*, oferind jucătorului posibilitatea de a-și construi și adapta strategia în funcție de disponibilitatea și utilizarea eficientă a resurselor disponibile. De exemplu, jucătorul poate alege să construiască mai multe turnuri de apărare de la începutul jocului, sau poate alege să construiască mai puține turnuri de apărare și să-și îmbunătățească structurile existente.

De regulă, pentru a stabili costul resurselor pentru turnurile de apărare, se pot folosii următoarele întrebări:

- Câți inamicii trebuie să eliminam pentru a obține resursele necesare pentru a construi un turn de apărare?
- Câți inamicii trebuie să eliminam pentru a obține resursele necesare pentru a îmbunătăți un turn de apărare?
- Câte resurse se pot obține în total la eliminarea unui val de inamici?

- Ce turnuri putem construi cu resursele obținute după ce eliminăm X valuri de inamici?
- Care este performanța unui turn în raport cu costul său?

Alte sugestii pentru a stabili costul resurselor pentru turnurile de apărare:

- Evaluați abilitățile și caracteristicile unice ale turnului în raport cu eficacitatea sa în respingerea inamicilor. Cu cât un turn oferă mai multe avantaje tactice și are un impact mai mare asupra valurilor de inamici, cu atât ar trebui să aibă un cost mai ridicat.
- Luați în considerare structura de dificultate a jocului și identificați momentele cheie în care jucătorul va avea nevoie de un anumit tip de turn de apărare. Dacă turnul este necesar pentru a face față unui val de inamici puternici sau pentru a rezolva anumite provocări speciale, costul său ar trebui să reflecte această importanță strategică.
- Monitorizați și analizați datele de joc, cum ar fi rata de utilizare a turnurilor, performanța acestora și comportamentul jucătorilor. Aceste informații pot oferi indicii despre ajustările necesare în costurile turnurilor.
- Includeți mecanisme de risc și recompensă în economie pentru a stimula luarea de decizii strategice. Oferiți oportunități jucătorilor de a investi resurse și de a obține recompense mai mari sau de a suporta pierderi în cazul unor alegeri nepotrivite.

Un alt mod în care poate fi folosit acest sistem este cel în generarea de valuri inamice. În loc ca tipurile de inamici să fie prestabilite, acestea pot fi alese în funcție de valoarea valului de inamici. Fiecare inamic având o valoarea, inamicii pot fi aleși aleatoriu până ajungem la valoarea totală de resurse pe care dorim s-o avem pentru valul respectiv.

Presupunem că avem următoare costuri pentru turnurile de apărare din Tabelul 3 și următoarele recompense pentru eliminarea inamicilor din Tabelul 4.

Turn de apărare pasiv	Cost construire
Atac bonus	100
Încetinire	50
Îngheț	300

Tabelul 3: Costul turnurilor de apărare pasive

Tip inamic	Recompensă
Simplu	10
Rapid	5
Durabil	30

Tabelul 4: Recompensă eliminare inamici

Daca valul 1 de inamici conține următoarea compoziție: 7 inamicii simpli, 2 rapizi și un durabil, atunci costul total al valului este de 7 * 10 + 2 * 5 + 1 * 30 = 115. Dacă jucătorul elimină acest val de inamici, atunci va primi 115 resurse. Cu suma aceast el si-ar permite să construiască un turn pasiv de tipul *Atac bonus* sau două turnnuri pasive de tipul *Încetinire*. Pentru a construi un turn de tipul *Îngheț* ar mai avea nevoie de 185 resurse. Aceastea pot fi obținute din următoarele două valuri.

3. Implementarea sistemelor

O parte importantă al oricărei idei de joc este implementarea acesteia. În capitolele următoare vom descrie implementarea sistemelor principale din care va fi compus jocul care se folosește de sistemul de jetoane de acțiune.

Pentru partea de implementare într-un motor de joc, vom ține cont de următoarele aspecte:

- Modularitate: dorim ca sistemele să nu fie foate cuplate între ele, astfel încât să putem schimba un sistem fără a afecta alte sisteme.
- Extensibilitate: dorim ca sistemele să fie ușor de extins, adaugarea sau eliminarea unei noi functionalități să nu afecteze în mod critic celelalte componente.
- Performanță: dorim ca jocul să ruleze la o rată de cadre pe secundă cât mai mare – de preferat cel puțin 60 FPS.
- Integrare: componentele trebuie să fie ușor de integrat fără a necesita modificări majore în codul existent.

Pentru modularitate și extensibilitate este recomandată o arhitectură bazată pe entități indentificabile prin etichete și controlate de sisteme centrale: fiecare turn de apărare, inamic și proiectil va avea o etichetă prin care va fi identificat, iar acestea vor fi coordonate de un sistem pentru turnuri, proiectile și altul pentru inamicii.

Acestea funcționează astfel: un inamic cu eticheta `e-1` trece pe langă un turn de apărare cu etichetă `t-1`; sistemul de detecție pentru turn înregistrează inamicul `e-1` ca fiind in proximitate. `e-1` devină noua țintă pentru `t-1`. `t-1` preia coordonatele lui `e-1` din sistemul central care coordonează toți inamicii. Odată preluate datele, `t-1` comunică cu sistemul de turnuri central să crează două proiectile cu etichetele `p-1` și `p-1`. Odată create, aceastea sunt înregistrate în sistemul de coordonare al proiectilelor. Aceastea se îndreaptă spre `e-1`, iar la coliziune se întamplă doua lucruri: `e-1` raportează sistemului central că acesta s-a lovit de proiectilul `p-1`, sistemul central preia datele proiectilului (în acest caz valoarea pagubei) și actualizează viața lui `e-1`. Proiectilul `p-1` raportează sistemului central de proiectile că s-a ciocinit de un inamic, prin urmare proiectilul trebuie să fie eliminat. Sistemul central de proiectile elimină proiectilul `p-1` din lista de proiectile active.

Dupa cum putem observa, avem multe indirecții. Entitățile nu comunică direct între ele, ci prin intermediul sistemelor centrale. Acesta se aseamănă cu un sistem de evenimente unde entitățile sunt emițători de evenimente, iar sistemele centrale sunt ascultători de evenimente [4]. Acest lucru ne permite să avem o arhitectură modulară, unde sistemele centrale pot fi schimbate fără a afecta entitățile.

Dacă folosim un motor de joc precum Unity, avem la dispoziție obiectele de tip `ScriptableObjects` [5] care ne permite schimbarea de module și sistem într-un mod foarte convenabil întrucât nu necesită recompilarea codului și totul poate fi făcut din interfața de utilizator a motorului de joc.

Problema integrării componetelor (din punct de vedere al codului) poate fi rezolvată prin folosirea de *design patterns* – aceastea sunt o serie recomandări pentru structura codului astfel încât să fie ușor de înțeles și de extins. De exemplu, în cazul nostru, vom folosii *design pattern*-ul *Observer* [6] pentru a implementa sistemul de evenimente. Inamicii, proiectilele și turnurile vor fi subiecte, iar sistemele centrale vor fi observatori.

Alte astfel de *design patterns* [6] [7] care pot fi folosite în implementarea jocului sunt:

- *Singleton*: pentru sistemele centrale, avem nevoie de o singură instanță a acestora;
- *Factory*: pentru crearea de entități simple;
- *Builder*: pentru crearea de entități complexe;
- *State*: pentru a gestiona starea entităților;
- *Strategy*: pentru a gestiona comportamentul entităților;
- *Command*: pentru a gestiona comenzile utilizatorului;

- *Decorator*: pentru a adăuga noi comportamente entităților;
- *Flyweight*: pentru a reduce memoria folosită de entități;
- *Prototype*: pentru a clona entități;
- *Adapter*: pentru a adapta interfețe;
- *Facade*: pentru a ascunde detalii de implementare;
- *Mediator*: pentru a gestiona comunicarea între entități;
- *Memento*: pentru a salva starea entităților;
- *Proxy*: pentru a gestiona accesul la entități;
- *Iterator*: pentru a itera prin entități;
- *Visitor*: pentru a itera prin entități și a le modifica;
- *Composite*: pentru a crea structuri de date complexe;
- *Observer*: pentru a gestiona evenimente;
- *Chain of responsibility*: pentru a gestiona evenimente;

Toate tehniciile enumerate anterior ne ajută la pastrarea unui cod curat și ușor de înțeles. Dar acestea nu rezolvă și problema performanței.

Performanța depinde foarte mult de modul cum gestionăm memoria și procesorul. Mereu vom dorii să minimizăm memoria folosită și să reducem timpul de procesare. În funcție de problema întâmpinată, uneori este mai bine să folosim mai multe memorie pentru a salva timp de procesare (unele calcule se pot memora și refolosii pe viitor), alteori este mai bine să folosim mai puțină memorie și să folosim mai mult timp de procesare (unele structuri de date pot ocupa foarte mult spațiu și este mai eficient să calculăm de fiecare dată).

O arhitectură recomandată pentru acest gen de jocuri este cea bazată pe *Entity Component System* (ECS) [8]. Aceasta este o arhitectură care se bazează pe următoarele principii:

- Entități (*Entities*): sunt obiecte care nu au niciun comportament, acestea sunt doar un identificator unic.
- Componente (*Components*): sunt obiecte care conțin date, acestea nu au niciun comportament.
- Sisteme (*Sistems*): sunt obiecte care conțin comportament.

Entitatea este un obiect virtual care reprezintă un element distinct și autonom din joc sau aplicație. Aceasta poate fi orice entitate interactivă sau obiect în lumea virtuală, cum ar fi un personaj, un inamic sau un obiect de mediu. Entitatea în sine nu conține logica specifică, ci funcționează ca un container pentru componente.

Componentele reprezintă caracteristicile și comportamentul specific asociate unei entități. Ele conțin date și funcționalități care definesc aspecte specifice ale entității, cum ar fi aspectul vizual, fizica, inteligența artificială sau orice alt aspect al jocului. De exemplu, o entitate de tip „Jucător” poate avea componente precum „Randare” pentru afișarea grafică, „Fizica” pentru interacțiunea fizică sau „ComenziJucator” pentru gestionarea intrărilor de la jucător.

Sistemele sunt entități specializate care preiau entitățile care îndeplinesc anumite criterii și aplică logica specifică asupra componentelor acestora. Această separare permite o gestionare și o extensibilitate mai ușoară a logicii jocului.

Cu această tehnică avem următoarele beneficii:

- Alinierea memoriei: Prin stocarea entităților și componentelor într-un mod continuu în memorie, ECS optimizează alinierea datelor, ceea ce duce la acces mai rapid și mai eficient la acestea. Acest aspect este important în jocurile de Tower Defense, unde există un număr mare de entități și componente care trebuie accesate în mod constant.
- Procesare paralelă: faciliterea procesarea paralelă a entităților și componentelor pe mai multe nuclee de procesor. Aceasta permite jocului să beneficieze de puterea de calcul a sistemului în mod eficient, accelerând logica jocului și reducând posibilele blocaje sau întârzieri.

- Modularitate și optimizare specifică: împărțirea logică a jocului în sisteme specializate care gestionează diferite aspecte, cum ar fi inteligența inamicilor, detectarea coliziunilor sau afișarea graficii. Acest lucru facilitează optimizarea specifică a fiecărui sistem, concentrându-se pe partea relevantă.
- Reutilizarea componentelor: reducerea consumul de memorie și costurilor asociate cu crearea și distrugerea constantă a obiectelor, permițând jocului să ruleze mai eficient.
- Gestionarea dinamică a entităților: adăugarea, eliminarea și modificarea entităților și componentelor în timp real. Acest aspect este util în jocurile de Tower Defense, unde numărul și tipurile de entități pot varia pe parcursul jocului – adaptabilitatea și scalabilitatea în funcție de necesități, fără a afecta în mod semnificativ performanța.

Un dezavantaj la această arhitectură este că este destul de dificil de înțeles și de implementat. În plus, organizarea și gestionarea entităților și componentelor implică un overhead care poate afecta complexitatea și costurile dezvoltării. O implementare incorectă poate duce la o scădere a performanței. Motorul de joc Unity oferă suport pentru ECS prin pachetul *DOTS* [9].

3.1. Primele minute de joc

Inainte de a intra în detalii despre implementarea sistemului de jetoane de acțiune, vom descrie primele minute de joc. Acest lucru ne va ajuta să înțelegem mai bine cum funcționează jocul și cum trebuie jucătorul să interacționeze cu acesta.

Prima dată când pornim jocul vom vedea meniul principal. În acest meniu avem următoarele opțiuni:

- *New Game*: începe un joc nou.
- *Load Game*: încarcă un joc salvat.
- *Options*: setările jocului.
- *Exit*: închide jocul.

Când apăsăm pe butonul *New Game* se va încărca scena de joc. În această scenă avem următoarele elemente:

- Harta de joc: reprezintă zona de joc, aici se vor desfășura toate acțiunile jocului.
- Interfața pentru magazin: reprezintă meniul de unde putem cumpăra turnuri de apărare și putem vedea informații despre acestea.
- Interfața pentru statusul jucătorului: elemente vizuale care arată informații despre resursele acumulate, viața obiectivului care trebuie protejat, numărul valului de inamicii, timpul rămas până la următorul val de inamicii, etc.
- Interfața pentru statusul inamiciilor: elemente vizuale care arată informații despre inamicii care se află pe hartă, cum ar fi: viața, tipul de jetoane de acțiune deținute, armura, abilități, etc.
- Turnurile de apărare: elemente vizuale care reprezintă turnurile de apărare care au fost construite pe hartă și elementele conexe acestora, cum ar fi: raza de acțiune, proiectilele, etc.
- Inamicii: elemente vizuale care reprezintă inamicii care se află pe hartă.

Inainte să apăsăm pe butonul de start al sesiunii, trebuie să amplasăm primele turnuri de apărare. Avem mai multe opțiuni de plasare, de exemplu:

1. Începem cu un turn activ care are un proiectil simplu și o rată de atac medie. Iar ca să ne asigurăm că acesta va fi eficient, vom folosii turnurile pasive de încetinire și îngheț. În acest fel, vom încetini inamicii și vom îngheța inamicii care au un jeton de încetinire de rang 2. Acest lucru ne va permite să ne asigurăm că proiectilele turnului activ vor lovi inamicii înghețați.

2. Începem cu un turn activ cu rată mare atac si proiectil rapid. Ca să-l folosim la potențialul său maxim, vom folosii turnurile pasive de atac bonus și încetinire. Ne vom asigura că inamicii vor fi încetiniți înainte de a intra în raza de acțiune a turnului de atac bonus, astfel încât inamicul să ajungă să aibă un jeton de atac bonus de rang înalt. Astfel, proiectilele turnului activ vor avea un efect mai mare asupra inamicilor.

După ce am plasat primele turnuri de apărare, putem apăsa pe butonul de start al sesiunii. În acest moment, primul val de inamici va fi generat și va începe să se deplaseze pe traseu. În acest moment, putem observa dacă amplasarea turnurilor a fost corectă sau nu. Dacă inamicii sunt eliminați înainte de a ajunge la obiectiv, este un semn că am facut decizia corectă. Dacă inamicii ajung la obiectiv, atunci trebuie să ne gândim la o altă strategie.

Odată terminat valul, e timpul să ne folosim de resursele acumulate pentru a achiziționa noi turnuri de apărare sau pentru a îmbunătăți turnurile existente. Cu fiecare val care trece inamicii devin mai puternici și numărul lor crește, iar noi trebuie să ne adaptăm strategia de joc pentru a face față provocărilor care apar.

În funcție de ce turn am ales la început, încercăm să maxizăm potențialul acestuia, dar în același timp să ne gândim cum putem combina turnurile pasive pentru a obține jetoane mai puternice. Dacă am încerca să investim în turnurile pasive de atac bonus și explosiv, ne-ar putea ajuta cu valurile care au mulți inamicii cu viață mică. Dacă am încerca să investim în turnurile pasive de încetinire și îngheț, ne-ar putea ajuta cu valurile care au inamicii cu viață mare.

3.2. Sistemul de jetoane de acțiune

Acest sistem se ocupă de gestionarea jetoanelor de acțiune. Un jeton de acțiune reprezintă o acțiune care poate fi efectuată de unele turnuri de apărare și acesta este purtat de către inamicii. Un turn de apărare poate crea un jeton de acțiune care poate fi consumat de un alt turn de apărare prin intermediul inamicilor.

Propunem următoare structură pentru definirea unui jeton de acțiune:

- Tip efect: reprezintă tipul de efect pe care îl produce jetonul de acțiune.
- Valoare efect: reprezintă intensitatea efectului.
- Rang maxim: reprezintă rangul maxim pe care îl poate avea jetonul de acțiune.
- Rang curent: reprezintă rangul curent pe care îl are jetonul de acțiune.
- Durată: reprezintă durata de timp pentru care jetonul de acțiune este activ.
- Durata curentă: reprezintă durata de timp rămasă pentru care jetonul de acțiune este activ.
- Condiții de creere: reprezintă condițiile care trebuie îndeplinite pentru a crea un jeton de acțiune.

Tipul de efect reprezintă acțiune care se va produce atunci când jetonul de acțiune este consumat. Aceste acțiuni reprezintă funcționalități ale mecanicii de joc. Iată câteva exemple de tipuri de efecte:

- Scăderea parțială sau completă a vitezei de mișcare al inamicilor.
- Scăderea parțială sau completă a vieții inamicilor.
- Creșterea pagubelor primite de inamici din partea proiectilelor turnurilor.

Valoarea efectului reprezintă intensitatea efectului. Aceasta poate fi un număr întreg sau un procentaj. De exemplu, un jeton de acțiune care scade viteza de mișcare a inamicilor cu 50% are o valoare efect de 50%. Acest valori fie pot fi constante sau calculate prin intermediul unor funcții sau formule matematice. Exemplu:

$$v = \frac{r}{10}$$

, unde $r \in [0, 10]$ reprezintă rangul jetonului de acțiune și $v \in [0, 1]$ reprezintă valoarea efectului.

Rangul maxim reprezintă numărul maxim de jetoane de acțiune de același tip pe care un inamic le poate deține. Un jeton de acțiune de rang mai mare are un efect mai pronunțat decât unul de rang mai mic. Când un inamic este expus de mai mult ori atacului unui turn de apărare, rangul jetonului de acțiune asociat cu turnul de apărare crește.

Rangul curent reprezintă rangul curent pe care îl are jetonul de acțiune. Acesta poate fi mai mic sau egal cu rangul maxim. Rangul curent crește atunci când un inamic este expus de mai mult ori atacului unui turn de apărare asociat cu jetonul respectiv.

Durata reprezintă durata de timp pentru care jetonul de acțiune este activ. Aceasta poate fi un număr real. De exemplu, un jeton de acțiune care scade viteza de mișcare a inamicilor cu 50% pentru 5 secunde are o durată de 5 secunde. Această valoare poate fi constantă sau calculată prin intermediul unor funcții sau formule matematice.

Durata curentă reprezintă durata de timp rămasă pentru care jetonul de acțiune este activ. Aceasta este un număr real care este actualizat la un interval de timp dat. Când durata curentă ajunge la 0, jetonul de acțiune este eliminat.

Condiții de creere reprezintă condițiile care trebuie îndeplinite pentru a crea un jeton de acțiune. Aceste condiții iau considerare următoarele aspecte: tipul de effect și rangul său curent. De exemplu, un jeton de înghețare poate fi creat doar dacă inamicul are un jeton de încetinire de rang 2.

Cum acest sistem trebuiă să aibă o **implementare concretă într-un limbaj de programare** pentru a fi integrat într-un joc, vom folosii limbajul de programare Rust[10] pentru a descrie o posibilă implementare. Exemplele pot fi prezentate și sub formă de pseudocod, dar realizarea lor cu un limbaj de programare face ca totul să fie mai tangibil și chiar să facă parte din implementarea jocului.

Un exemplu concret de structură pentru un jeton de acțiune este următoarea:

```
struct Token {
  effect_type: EffectType,
  effect_value: float,
  max_rank: int,
  current_rank: int,
  duration: float,
  current_duration: float,
  creation_conditions: List<Condition>
}
```

Pentru tipul de efect putem avea un simplu *enum* care să conțină toate tipurile de efecte pe care le putem avea:

```
enum EffectType {
  Slow,
  Freeze,
  BonusAttack,
  Explosion
}
```

, iar pentru condiția de creere avem următoarea structură:

```
struct Condition {
  effect_type: EffectType,
  min_rank: int
}
```

3.3. Sistemul de turnuri de apărare

Acest sistem se ocupă de gestionarea turnurilor de apărare. Un turn de apărare reprezintă o structură care exercită o acțiune aupra inamciilor care duce în mod direct sau indirect la eliminarea acestora. În acest sistem, vom avea două tipuri de turnuri de apărare: turnuri active și turnuri pasive.

Turnurile active au următoarele caracteristici:

- Produc proiectile care pot elimina inamicii.
- Au o rată de atac proprie.
- Au sistem de țintire.
- Au rază de atac.

Proiectilele fac parte din mecanismul turnurile active, dar cum pot varia de la un turn la altul, vom avea o structură separată pentru acestea cu următoarele caracteristici:

- Viteză de mișcare: reprezintă viteza de deplasare a proiectilului.
- Pagube: reprezintă pagubele pe care le produce proiectilul.
- Durată de viață: reprezintă durata de timp pentru care proiectilul este activ.
- Durată de viață curentă: reprezintă durata de timp rămasă pentru care proiectilul este activ.

Turnurile pasive au următoarele caracteristici:

- Produc jetoane de acțiune care pot fi consumate de alte turnuri.
- Au rază de influență.
- Jetoanele se aplică la un interval de timp dat tuturor inamicilor din rază de influență.

Codul structurii pentru un turn de apărare activ și proiectil este următorul:

```
struct ActiveTower {
  projectile: Projectile,
  attack_rate: float,
  attack_rate_timer: float,
  range: float,
}

struct Projectile {
  speed: float,
  damage: float,
  life_time: float,
  current_life_time: float,
}
```

Pentru turnurile pasive avem următoarea structură:

```
struct PassiveTower {
  token: Token,
  range: float,
  token_rate: float,
  token_rate_timer: float,
}
```

Acest sistem este simplu de implementat, cea mai complicată partea find partea de țintire a turnurile active. In mod ideal, am dorii să trimitem către inamic un număr minim de proiectile care să-l elimine. Astfel, turnul devine mai eficient prin faptul ca nu pierde timp pentru a elimina un inamic care oricum va fi eliminat de către alt turn sau de proiectilele create precedent.

O exemplu de algoritm de țintire ar fi următorul:

1. Alegem ce mai apropiat inamic din raza de acțiune.

2. Calculăm câte proiectile sunt necesare pentru a-l elimina folosind punctele de viață și pagubele proiectilului.
3. Calculăm pozițiile inamicilor la momentul când aceștia vor fi loviți de proiectile.
4. Verificam dacă la pozițiile calculate vor lovii și proiectilele de la alte turnuri.
5. Dacă exista asemenea caz, recalculăm numărul de proiectile tinând cont de punctele de viață rămase după ce inamicul este lovit de proiectilele de la alte turnuri.

O problemă care acest algoritm nu o ia in considerare este efectul provocat de jetoanele de acțiune asupra inamicilor. Și anume ca inamicul nu va avea o viteză de mișcare constantă pe traseu, iar acesta poate primii pagube de la inamicii învecinații care au asupra lor jetoane de explozie care îl pot elimina înainte ca proiectilele să ajungă la el. Acest scenariu este prea complex pentru un simplu joc dar reprezintă un caz interesent de cercetare.

Pentru turnurile pasive, nu e nevoie de niciun algortim de țintire, acestea având un efect asupra tuturor inamicilor din rază de influență. Există un timer care la un interval de timp dat, toate turnurile pasive aplică jetoanele de acțiune.

3.4. Sistemul de economie de joc

Acest sistem se ocupă de gestionarea resurselor. Resursele sunt folosite pentru a construi turnuri de apărare și pentru a le îmbunătăți. Resursele pot fi obținute prin intermediul unor structuri speciale sau pot fi obținute prin eliminarea inamicilor.

În jocuri, acestea au denumiri generice precum: *gold* (aur), *coins* (monede), *gems* (nestemate), etc. Numărul lor variază în funcție de joc, iar unele jocuri pun mai mare accent pe gestionarea resurselor decât altele.

În principal, acest sistem nu are un impact prea mare asupra mecanii de joc la un joc Tower Defense clasic, dar prin combinarea mai multor genuri de jocuri, se poate crea ceva unic. De exemplu, în jocuri populare care se combină cu *Role Play Game* (RPG), această parte devine fundamentală pentru joc. Un astfel de joc este *Arknight*[11] care combină genul Tower Defense cu RPG, unde jucătorul trebuie să colecteze o varietate de resurse pentru a avansa în joc. Si cum este un proces încet, se pot folosii bani reali pentru a cumpăra resurse – aceasta find și principala sursă de monetizare a jocului.

Un tip de resursă poate fi definit astfel:

```
enum ResourceType {
  Gold,
  Wood,
  Stone,
}
```

Atunci când eliminăm un inamic, primim o recompensă sub formă de resurse. Această recompensă compusă din tipul de resursă și suma. De exemplu, un inamic poate să ofere 10 aur, 5 lemn și 2 piatră când este eliminat. O structură pentru recompensă poate fi definită astfel:

```
struct Reward {
  resource_type: ResourceType,
  amount: float,
}
```

Stiind câți inamicii avem într-un val și care este valoarea, putem calcula recompensa totală pentru valul respectiv. De exemplu, dacă avem 10 inamicii care oferă 10 aur, 5 lemn și 2 piatră, recompensa totală pentru valul respectiv este de 100 aur, 50 lemn și 20 piatră. Cu această informație putem stabilii costul resurselor pentru turnurile de apărare. De exemplu, un turn de apărare poate costa 20 aur, 10 lemn și 4 piatră. Deci cu resursele pe un întreg val putem achiziționa 5 turnuri de apărare.

În ultimii ani au devenit din în ce mai populare jocurile de tip *base building*, în care scopul jocului e să construiești clădiri care au nevoie de un lanț de aprovizionare complex. Acestea constau dintr-un mare de resurse care trebuie să fie colectate și gestionate. Multe dintre aceste fiind combinații de alte resurse, exemplu: pentru a produce o căramidă avem nevoie o fabrică și o mină de clei care să extraga resursa de bază, și anume cleiul. Practic, avem nevoie de două fabrici și o resursă de bază pentru a produce resursa compusă.

Acest lanț de aprovizionare seamănă foarte mult cu idea noastră de jetoane de acțiune. Pentru a produce un jeton de îngheț trebuie să avem un turn pasiv de înghețare care trebuie să primească un jeton de încetinire de rang 2, produs la rândul său de un turn pasiv de încetinire.

Prin urmare, jocurile de tip *base building* pot fi considerate repere pentru implementarea sistemului de jetoane de acțiune, având în vedere asimilările.

3.5. Sistemul de inamicii

Inamicii reprezintă entitățile care trebuie să ajungă la obiectivul pe care jucătorul încearcă să-l protejeze. Pentru a ajunge la destinație, acesta trebuie să supraviețuiască atacurilor turnurilor de apărare. Inamicii au o viață și o viteză de mișcare. Viața reprezintă numărul de pagube pe care un inamic le poate suporta înainte de a fi eliminat. Viteza de mișcare reprezintă cât de repede se deplasează inamicul pe traseu.

Un inamic are următoarele caracteristici:

- Viață: reprezintă numărul de pagube pe care le poate suporta inamicul înainte de a fi eliminat.
- Viteză de mișcare: reprezintă cât de repede se deplasează inamicul pe traseu.
- Jetoane de acțiune: reprezintă jetoanele de acțiune pe care le deține inamicul.
- Recompensă: reprezintă recompensa pe care o oferă inamicul când este eliminat.

În baza acestor caracteristi putem avea următoarele tipuri de inamicii:

- Inamic rapid: acesta are o viață mică și o viteză de mișcare mare.
- Inamic normal: acesta are o viață medie și o viteză de mișcare medie.
- Inamic rezistent: acesta are o viață mare și o viteză de mișcare mică.

La primă vedere pare să nu avem prea multe opțiuni de variație pentru inamicii. Unele jocuri de Tower Defense încearcă să crească acest număr prin introducerea unor noi mecanici de joc pentru inamicii, precum:

- Rezistență la anumite tipuri de atac. Tunurile de apărare pot avea diferite tipuri de atac, iar inamicii pot avea rezistență la un anumit tip de atac. De exemplu, un inamic poate avea rezistență la atacurile de foc, iar turnurile de apărare pot avea atac de foc.
- Regenerare viață.
- Imunitate la anumite efecte date de către turnuri.
- Abilități speciale. Exemplu: poate devenii invulnerabil pentru o perioadă de timp; poate crește viteze de mișcare a inamicilor din jurul său;

Structura unui inamic poate fi definită astfel:

```
struct Enemy {
  health: float,
  speed: float,
  tokens: List<Token>,
  reward: Reward,
}
```

3.6. Interfața de utilizator

Interfața de utilizator joacă un rol critic în experiența de joc, deoarcere sistemul de jetoane de acțiune este un sistem complex, iar jucătorul trebuie să aibă o interfață intuitivă pentru a putea interacționa cu acesta.

Avem următoarele elemente de interfața de utilizator trebuie să le afișeze:

- informații despre turnurile de apărare.
- informații despre inamicii care se află pe hartă.
- informații despre statusul jucătorului.
- informații despre statusul inamicilor.
- informații despre resursele jucătorului.
- informații despre obiectivul care trebuie protejat.
- informații despre valul de inamicii.
- informații despre timpul rămas până la următorul val de inamicii.
- informații despre turnurile de apărare care se află pe hartă.

În Figura 3 putem obsera un exemplu pentru creearea unor zone în care putem pune elementele vizuale pentru afișarea informațiilor.

Și următoarele reguli pentru construirea interfeței de utilizator [12], [13]:

- informațiile trebuie să fie afișate într-un mod intuitiv, clar și concis.
- elementele nu trebuie să distragă atenția de la joc.
- trebuie să existe o ierarhie a importanței între elemente vizuale.
- estetica elementelor trebuie să fie în concordanță cu tema jocului.

Magazinul (*shop*) este un element important al interfeței de utilizator. Acesta este locul unde putem cumpăra turnuri de apărare și putem vedea informații despre acestea. Acesta are trei secțiuni: turnuri de apărare active, turnuri de apărare pasive și îmbunătățiri generale.

Pentru turnurile de apărare active, avem următoarele informații:

- numele turnului de apărare.
- imaginea turnului de apărare.
- descrierea caracteristicilor.
- costul de achiziție.
- butonul de cumpărare.

Figura 4 reprezintă o schiță pentru un astfel de element vizual. Pentru turnurile de apărare pasive, avem următoarele informații:

- numele turnului de apărare;
- descrierea caracteristicilor;
- costul de achiziție;
- butonul de cumpărare;
- condițiile de creere pentru jeton;
- jetonul de acțiune creat.

Figura 5 reprezintă o schiță pentru un element vizual al unui turn pasiv din magazin. Iar pentru îmbunătățiri, avem:

- numele îmbunătățirii;
- descrierea;
- costul de achiziție;
- butonul de cumpărare;

- tipul de turn de apărare va fi îmbunătățit;
- caracteristicile care vor fi îmbunătățite.

Figura 6 este schița pentru un astfel de element vizual.

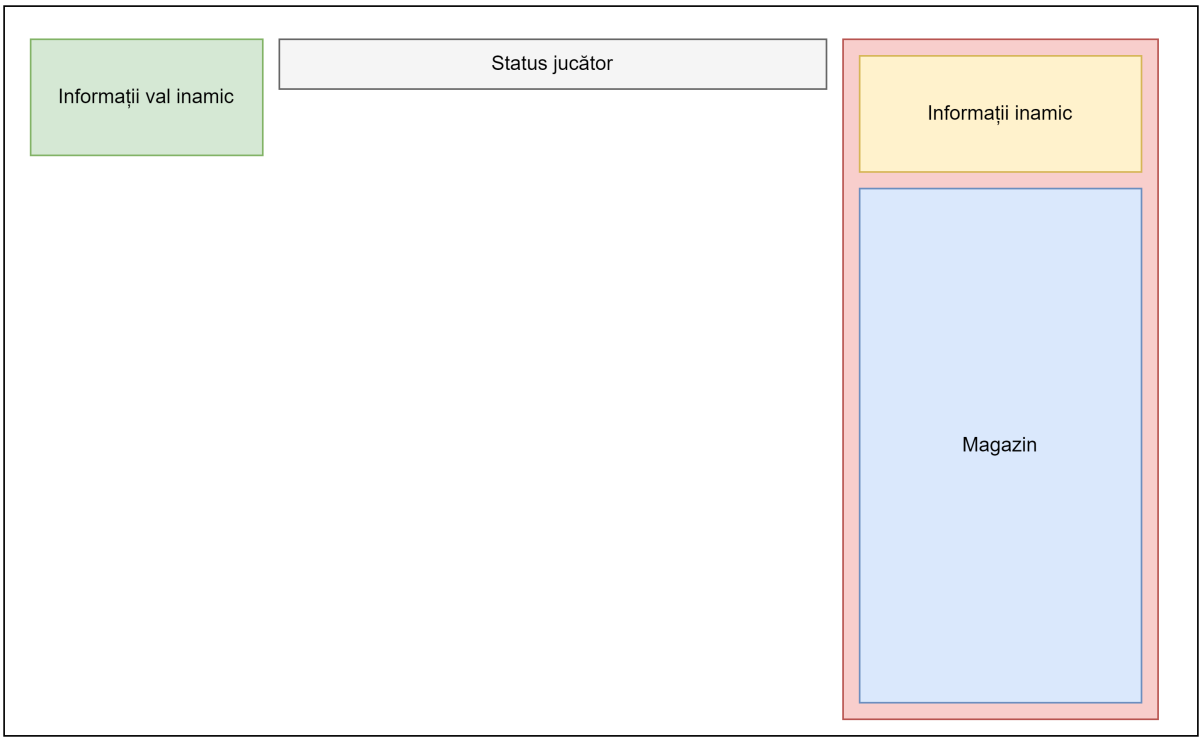

Figura 3: Schiță pentru interfața de utilizator.

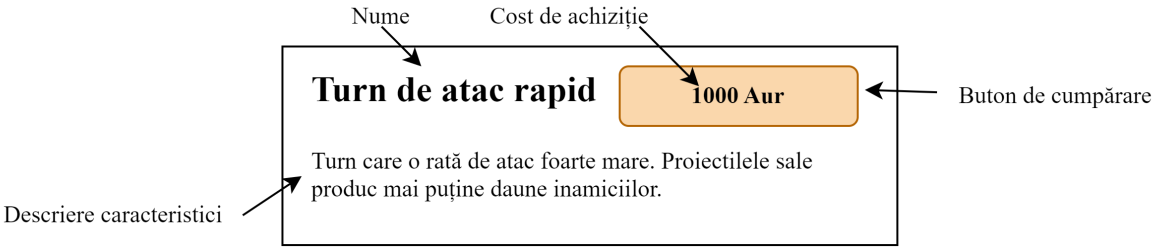

Figura 4: Schiță pentru un element vizual al unui turn activ din magazin.

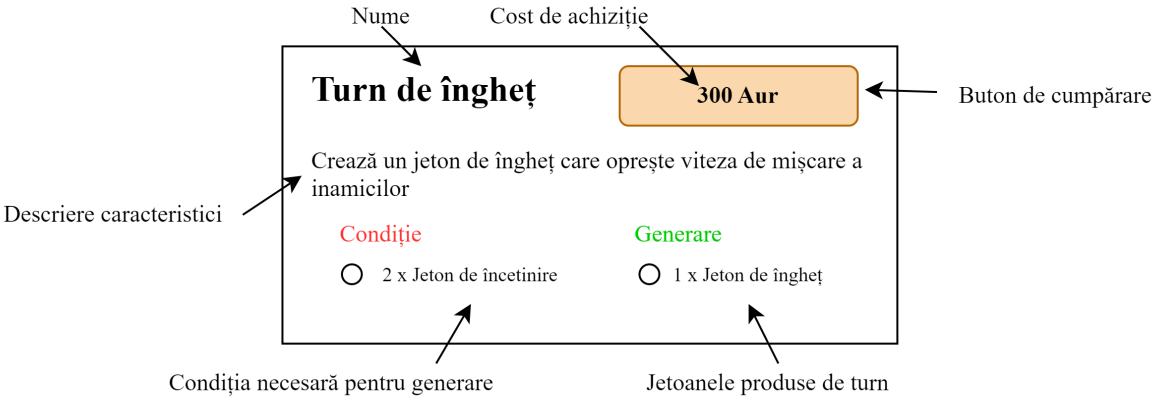

Figura 5: Schiță pentru un element vizual al unui turn pasiv din magazin.

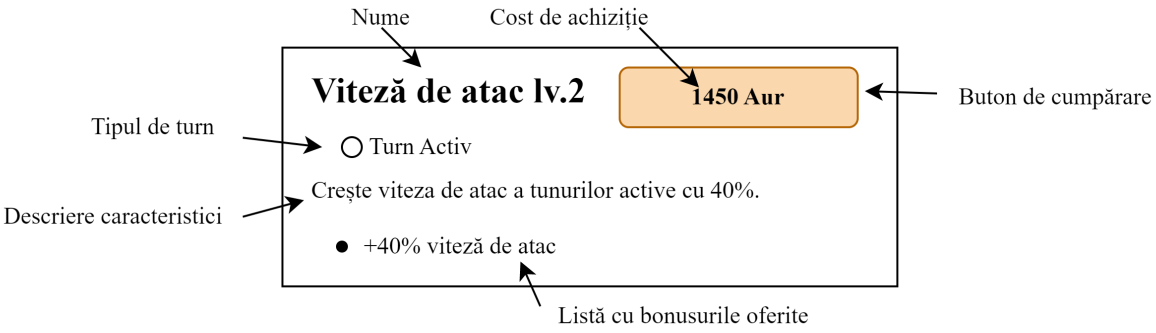

Figura 6: Schiță pentru un element vizual al unui îmbunătățiri din magazin.

Bibliografie

[1] A. Hernández-Sabaté, M. Joanpere, N. Gorgorió, and L. Albarrac\in, “Mathematics learning opportunities when playing a tower defense game,” *Int. J. Serious Games*, vol. 2, no. 4, pp. 57–71, 2015.

[2] P. Avery, J. Togelius, E. Alistar, and R. P. Van Leeuwen, “Computational intelligence and tower defence games,” in *2011 IEEE Congr. Evol. Computation (Cec)*, 2011, pp. 1084–1091.

[3] “Game design principles - tower defense.” [Online]. Available: https://www.reddit.com/r/gamedesign/comments/j70hme/game_design_principles_tower_defense/

[4] J. Walcher, “Event-driven game engine in realtime strategy games.”

[5] “Three ways to architect your game with scriptableobjects.” [Online]. Available: https://unity.com/how-to/architect-game-code-scriptable-objects

[6] E. Gamma, R. Helm, R. Johnson, R. E. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Pearson Deutschland GmbH, 1995.

[7] R. Nystrom, *Game Programming Patterns*, Genever Benning, 2014.

[8] T. Härkönen, “Advantages and implementation of entity-component-systems,” 2019.

[9] “Unity’s data-oriented technology stack (dots).” [Online]. Available: https://unity.com/dots

[10] “Rust.” [Online]. Available: https://www.rust-lang.org/

[11] “Arknight wiki.” [Online]. Available: https://arknights.fandom.com/wiki/Arknights_Wiki

[12] “Best practices for designing an effective user interface.” [Online]. Available: https://www.gamesindustry.biz/best-practices-for-designing-an-effective-video-game-ui

[13] *User Interface Design and Implementation in Unity*, Unity Technologies, 2022. [Online]. Available: https://resources.unity.com/games/user-interface-design-and-implementation-in-unity