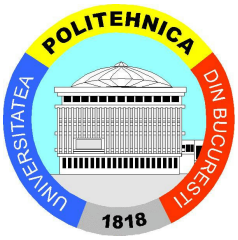
UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL DE CALCULATOARE



PROIECT DE DIZERTAȚIE

Design pentru genul de joc Tower Defense

Soare Robert Daniel

Coordonator Științific:
Prof. Dr. Ing. Moldoveanu Alin

BUCUREȘTI
2023

Cuprins

SINOPSIS

Jocurile sunt una dintre cele mai înalte forme de artă, combinând elemente de design, muzică, grafică și poveste pentru a crea o experiență unică. Jocurile video au devenit o parte importantă a culturii moderne, constituind o formă de divertisment foarte populară în rândul oamenilor de toate vârstele. Popularitatea lor se datorează modului de interactivitate pe care îl oferă, permițând jucătorilor să se implice în poveste și să ia decizii care influențează desfășurarea jocului.

Crearea unui design de joc presupune multă creativitate și o înțelegere profundă a mecanicilor de joc. Un joc de succes trebuie să fie captivant, să ofere o experiență de joc plăcută și să fie ușor de înțeles pentru cei care îl joacă. În această lucrare, ne propunem să creăm un design pentru un joc de tip *Tower Defense*, care să ofere o experiență nouă și interesantă. Exemplificarea acestui design va fi realizată prin implementarea unei aplicații interactive care demonstrează mecanica de joc propusă. Această lucrare are rolul de a inspira și de a oferi o bază pentru cei care doresc să creeze jocuri de tip *Tower Defense*.

ABSTRACT

Games are one of the highest forms of art, combining elements of design, music, graphics, and story to create a unique experience. Video games have become an important part of modern culture, serving as a highly popular form of entertainment among people of all ages. Their popularity is due to the level of interactivity they offer, allowing players to engage in the story and make decisions that influence the game's progression.

Creating a game design requires a lot of creativity and a deep understanding of game mechanics. A successful game must be captivating, provide an enjoyable gaming experience, and be easy to understand for those who play it. In this work, we aim to create a design for a Tower Defense type game that offers a new and interesting experience. The demonstration of this design will be carried out by implementing an interactive application that demonstrates the proposed game mechanics. This work aims to inspire and provide a basis for those who wish to create Tower Defense type games.

1. INTRODUCERE

Genul *Tower Defense* (sau „apărare prin turnuri”) este un gen de joc video din categoria jocurilor de strategie în care jucătorul are rolul de a construi și de a îmbunătății turnuri de apărare pentru a împiedica invazia inamicilor și a proteja un obiectiv. 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 ajungă la obiectiv. 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.

Jocurile moderne de *Tower Defense* au început să apară în anii 1990, iar popularitatea lor a crescut odată cu apariția modurilor de joc pentru jocul *Warcraft 3: The Frozen Throne* [1]. Perioada 2007-2012 a fost perioada de aur a acestui gen de joc, în care au fost lansate multe jocuri de succes, cum ar fi: *Plants vs. Zombies*, *Kingdom Rush*, *Bloons TD 6*, *Orcs Must Die!*, *Dungeon Defenders* și *Factorio*. Aceste jocuri au fost lansate pe diferite platforme: calculatoare personale, console și dispozitive mobile.

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. Această experiență poate varia de la o valoarea educativă [2] la una pur distractivă.

Acest gen de joc este tot mai întâlnit pe platformele de jocuri mobile, precum *Google Play* și *App Store*, unde generează venituri substanțiale [3]. Punctele forte 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 explicit al unui alt turn pentru a elimina inamicii.

Deoarece mecanica de joc moștenită este similară cu cea a modurilor din jocul *Warcraft 3: The Frozen Throne*, (moduri precum: *Element TD*), putem considera *Tower Defense* ca find un subgen al jocurilor de strategie în timp real. Dar față de un joc de strategie în timp real, îi lipsește partea de competitivitate din prisma faptului ca nu avem oponenți umanii și unitățile inamice nu au o strategie de joc.

Această lipsă a laturii competitive face ca mecanica de joc să fie monotonă și să nu ofere o experiență de joc variată pe o perioadă lungă de timp. Mulți dezvoltatori încearcă să rezolve

această problemă prin combinarea cu un alt gen de joc, cum ar fi: jocuri de rol, jocuri de acțiune. Dar toate acestea nu schimbă mecanica moștenită a jocului, ci doar o extind. Pentru a scăpa de această monotonie trebuie să reimaginăm mecanica de joc, dar ca să facem acest lucru trebuie să înțelegem mai bine mecanica de joc a jocurilor de *Tower Defense*.

Jocurile de strategie pot fi catalogate în funcție de modul în care jucătorii interacționează cu jocul. În general, jocurile de strategie se împart de-a lungul unui spectru în care capetele sunt reprezentate de: jocuri de strategie în timp real și jocuri de strategie pe ture. Jocurile în timp real se bazează pe reflexe rapide în luarea deciziilor strategice, iar cele pe ture se bazează pe planificare și analiză. Jocurile de *Tower Defense* au istorie în a fi derivate din jocurile de strategie în timp real, dar cum mecanica de joc nu necesită reflexe rapide, acestea se aseamană cu cele dinspre categoria jocurilor de strategie pe ture. Ca un punct de plecare, ne puteam orienta să schimbam mecanica de joc actuală cu una care seamană mai mult cu cea a jocurilor de strategie pe ture.

Un punct de insipirație sunt jocurile de cărți (poker, Război, *Uno*, *Magic: The Gathering*, *Yu-Gi-Oh!*, *HearthStone*) și șah-ul, care au un arbore de joc (*game-tree complexity*) foarte mare – șahul are o complexitate de 10^{123}[4]. În acestea, cartea sau piesa de joc individuală nu are un mare impact asupra jocului, cea ce contează este combinație/secvența de cărți/piese. Putem aduce această idee și în jocurile de Tower Defense, unde turnurile de apărare nu au un impact major asupra jocului în mod independent, ci combinația de turnuri de apărare este cea care contează. Astfel jocul o să aibă o latură combinatorică mai pronunțată în mecanica de joc [5].

În cadrul acestei lucrări, intenționăm să explorăm un model semi-interdependent de colaborare pentru turnurile de apărare. Acest model se va baza pe un sistem de comunicare prin mesaje pentru interacțiunea dintre turnuri. Aceste mesaje, denumite **jetoane de acțiune**, vor fi transportate de către inamici. Astfel, turnurile vor iniția acțiuni declanșate de aceste jetoane când inamicii se află în raza lor de acțiune, iar rolul jucătorului va fi să se asigure că aceste jetoane ajung la turnurile corespunzătoare pentru a activa 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. Scopul principal al acestor specificații este să motiveze dezvoltatorii de jocuri în a explora noi posibilități în mecanica de joc.

2. 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 îmbunătățească 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 îmbunătățească, pentru a se asigura că pot gestiona cu succes amenințările inamicilor.

Genul de joc Tower Defense își are rădăcinile într-un joc numit „Maze Games" care a început să fie jucat în anii 1980. În aceste jocuri, jucătorii trebuiau să-și construiască un traseu de tip labirint 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 viitoarele titluri care urmau să apară [6].

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 alt moment liber.
- Implică planificarea, gestionarea resurselor și luarea deciziilor strategice pentru a proteja o anumită zonă de atacurile inamicilor.

Alte motivații pot fi [7]:

- 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 ale acestui gen de joc care au aparut de-a lungul 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 moduri unice de a ataca. Jucătorii trebuie să le plaseze strategic pentru a opri zombii înainte ca aceștia să ajungă la obiectiv. Inamicii vin de la dreapta la stânga pe 5 rânduri. Fiecare rând reprezintă traseul unui zombi, iar o plantă poate ataca doar zombii de pe același rând cu aceasta. Acest a fost dezvoltat de către PopCap Games și a fost lansat în anul 2009.
- Kingdom Rush – Acest joc strategic în timp real plasează jucătorii într-o lume de basm unde aceștia își construiesc și își organizează turnurile defensive și trupele pentru a opune

rezistență și a învinge hoardele de creaturi mitice. Fiecare turn și trupă dispune de abilități și caracteristici unice, creând o gamă variată de strategii pe care jucătorii le pot utiliza. De asemenea, jucătorii au la dispoziție o serie de abilități speciale, care pot fi folosite pentru a influența în mod direct desfășurarea luptei. Jocul oferă o varietate de niveluri, fiecare cu propriile provocări și scenarii, cerând astfel o adaptare continuă a strategiilor de joc. Acesta 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 simplă in care trebuie sa apare baza de baloane, jucătorii trebuie să își plaseze turnurile defensive (care au forma unor maimuțe) și să își upgradeze abilitățile pentru a împiedica baloanele să ajungă la final. Acesta dispune de un număr mare de hărți de joc care prezintă diverse provocări prin prisma formei lor. 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. Acesta 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 că 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 – În acest joc hibrid de tipul Tower Defense și RPG (*Role-playing game*), jucătorii trebuie să își asume roluri de diferite clase de caractere, fiecare cu abilități și puncte forte unice, pentru a proteja cristalul Eternia de hoardele de inamici. Fiecare clasă de personaj poate construi și îmbunătăți turnuri de apărare, capcane și bariere, pentru a opri inamicii înainte ca aceștia să ajungă la cristal. De asemenea, jocul permite și implicarea directă a personajelor în luptă. Acesta oferă o gamă largă de niveluri și moduri de joc, având o componentă de progresie puternică care permite îmbunătățirea și personalizarea abilităților personajelor și a echipamentelor. *Dungeon Defenders* poate fi jucat atât în mod *single player* (un singur jucător) cât și *multiplayer* (mai mulți jucători), introducând astfel un element de colaborare și strategie de echipă. Jocul a fost lansat în anul 2010 și a fost dezvoltat de către Trendy Entertainment.
- Factorio – Joc de strategie în timp real, jucătorii trebuie să construiască și să mențină automate industriale într-o planetă extraterestră. Resursele sunt esențiale și jucătorii trebuie să colecteze și să le proceseze pentru a construi mașinării industriale complexe într-un mod eficient. Un alt element central al jocului este rețeaua logistică sofisticată, în care lanțul de aprovizionare al clădirlor trebuie atent planificat pentru a obține maximul de resurse. Jucătorii trebuie, de asemenea, să se apere de creaturile ostile ale planetei. Jocul a fost lansat în anul 2016 și a fost dezvoltat de către Wube Software.

Fiecare joc de Tower Defense are propriile sale particularități care le dau unicitate. Aceste particularități pot fi observate în diferite aspecte ale jocului, cum ar fi: grafica, povestea, mecanica de joc, modul de progresie, modul de joc, etc.

3. DESIGN-UL UNUI JOC NOU DE TOWER DEFENSE

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

- Obiective care trebuie apărate de atacurile inamicilor.
- O hartă de joc cu un traseu liber sau prestabilit.
- Inamici care încearcă să ajungă la obiective care trebuiesc protejate.
- Turnuri defensive care trebuie plasate strategic pentru a opri inamicii înainte ca aceștia să ajungă la obiective.
- Resurse care pot fi colectate pentru a construi turnurile defensive.

O reprezentare simplificată poate fi observată în Figura 1. Toate aceste componente trebuie să creeze experiența de joc caracteristică acestui gen de joc: crearea unui sistem de apărare care să poată rezista invadatorilor.

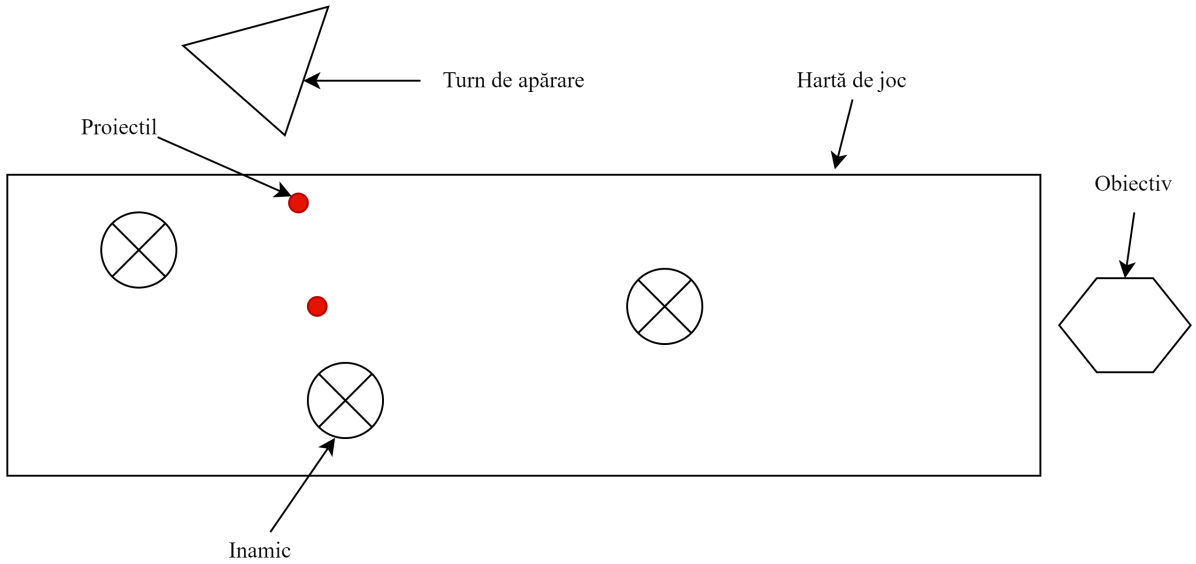


Figura 1: Schiță pentru un element vizual al unui îmbunătățiri din magazin.

Alte aspecte care pot fi luate în considerare în design-ul unui joc de Tower Defense sunt [7] [8][9]:

- Pagubele cauzate de turnuri asupra inamicilor - fiecare inamic are un anumit număr de puncte de viață, iar turnurile au un anumit număr de proiectile care sunt create într-un interval de timp. 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 țintire - 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 obiectiv, 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 ale 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 al mulțimii - unele turnuri pot avea efecte de control care se aplică unui număr mare de inamicii. 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.

În timp ce conceptul de bază al jocurilor de *Tower Defense* rămâne relativ simplu, există o mulțime de direcții și evoluții posibile în ceea ce privește designul acestor jocuri. Acest lucru le face să fie atrăgătoare atât pentru dezvoltatori, cât și pentru jucători, care se pot bucura de o varietate de abordări și mecanici în această categorie de jocuri. Design-ul jocurilor de *Tower Defense* a evoluat semnificativ în ultimii ani. Iată câteva exemple de arii 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 hărți 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 gestionare a resurselor (*resource management game*).
- O altă tendință în design-ul jocurilor de Tower Defense este implementarea de elemente RPG. Turnurile pot acum câștiga experiență și evolua, oferind jucătorilor un sentiment de progresie și un motiv să continue să joace.
- În ceea ce privește designul inamicilor, o evoluție importantă a fost introducerea de „șefi” sau inamici speciali care au capacitatea de a schimba dinamica jocului. Acești inamici puternici nu numai că oferă un nivel mai mare de dificultate, dar încurajează și diversitatea în plasarea și tipul turnurilor folosite de jucători.

- Mecanica de poveste sau narațiunea încorporată în jocurile de Tower Defense a devenit tot mai complexă. Prin introducerea de poveste și personaje, jucătorii pot fi mai angajați și investiți emoțional în joc, aducând un plus de valoare sesiunilor de joc.

Evoluțiile menționate reprezintă doar câteva din direcțiile în care acest gen de jocuri s-a dezvoltat. Chiar și cu aceste schimbări, esența jocului de Tower Defense rămâne aceeași: construirea unei defensive puternice pentru a împiedica avansarea inamicilor. Însă, noile elemente de gameplay și varietatea de opțiuni disponibile pentru jucători adaugă complexitate și oferă un grad mai mare de libertate în strategie, asigurând în același timp că fiecare experiență de joc este unică și provocatoare.

3.1. Modele hibride de jocuri

Cele mai bune jocuri sunt de multe ori cele care reușesc să îmbine elemente din mai multe genuri de jocuri pentru a crea o experiență de joc unică. Dar acest nu este un lucru ușor de realizat, deoarece trebuie să avem în vedere proporțile în care sunt combinate elementele din fiecare gen de joc. În multe jocuri populare din ultimii ani [10][11], *Tower Defense* este prezent în proporții destul de mici, deoarece este folosit ca un element de *gameplay* secundar. Design-ul simplist și ușor de înțeles face ca integrarea să fie relativ ușoară, iar jucătorii pot fi atrași de acest element de *gameplay* fără a fi nevoie să depună un efort considerabil în a învăța aceste elemente.

Unul din scopurile secundare al noilor idei de design care vor fi discutate în această lucrare este de a oferi oportunități și nu piedici în a crește prezența genului *Tower Defense* în modele hibride de jocuri. Iată câteva exemple de genuri de joc care sunt îmbinate de multe ori cu *Tower Defense*:

- 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 forma unei provocări

(*challenge mode*) care poate fi jucate de către jucători după ce au terminat părțile principale. Aceasta constă în adaugarea de noi constrăngeri 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 se concentrează pe elaborarea și implementarea unei strategii în timp real. De cele mai multe ori, aceste jocuri au un component competitiv bine definit, unde orice greșeală poate fi exploatată. Majoritatea jocurilor din acest gen includ turnuri de apărare ce au rolul de a încetini avansul adversarilor.

Modelele hibride de Tower Defense îmbogățesc experiența jucatorilor prin combinarea elementelor caracteristice altor genuri de jocuri. Astfel, jucătorii pot descoperi noi abordări strategice, pot experimenta diverse scenarii și sunt provocați să gândeasca în mod creativ pentru a-și asigura victoria. Modelul de jetoane de acțiune nu împiedică implementarea acestor modele hibride, ci dimpotrivă, îi oferă un plus de valoare prin introducerea unui nou element de strategie. Provocarea dezvoltatorilor de jocuri este de a crea un sistem de jetoane de acțiune care să fie flexibil și să poată fi adaptat la orice model hibrid de joc fără a crea redundanță sau conflicte în experiența de joc.

De asemenea, este important de menționat că, în timp ce introducerea de noi elemente și modele hibride poate adăuga un grad de complexitate, acest lucru nu trebuie să fie văzut neapărat ca un dezavantaj. Dimpotrivă, un nivel adecvat de complexitate poate stimula gândirea strategică a jucătorilor și poate asigura o durată de viață mai lungă a jocului. Totuși, este crucial ca acest nivel de complexitate să fie gestionat cu atenție, pentru a evita crearea unui joc care să fie perceput ca fiind prea greu sau confuz.

Așadar trebuie să avem în vedere următoarele aspecte când dorim să integram un nou element în design-ul jocului:

- Echilibrul *gameplay*-ului: Orice element nou trebuie evaluat pentru a preveni perturbarea armoniei în joc. Componente supra sau subevaluate pot dăuna experienței utilizatorului.
- Progresia jucătorului: Dacă adăugăm elemente noi, acestea ar trebui introduse treptat, permițând jucătorului să învețe și să se adapteze. Evităm să încărcăm jucătorul cu prea multe concepte noi simultan.
- Testare și *feedback*: Înainte de a finaliza orice nouă implementare, este esențial să testăm intensiv și să solicităm păreri de la jucători. Acest proces ne ajută să identificăm potențialele probleme și să îmbunătățim experiența de joc înainte de lansarea finală.
- Coerență în design: Importul elementelor din alte genuri trebuie făcut într-un mod care să păstreze esența originală a jocului de Tower Defense. De pildă, dacă adăugăm componente din jocurile RPG, trebuie să ne asigurăm că acestea nu minimalizează importanța turnurilor de apărare.

Ținând cont de aceste idei exprimate anterior, putem începe să explorăm posibilitățile de a extinde mecanica de joc a jocurilor de Tower Defense fără a crea impedimente în partea creativă și de design.

3.2. 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. Aceste acțiunii pot fi de diferite tipuri, cum ar fi:

- Bonus atac: inamicul primesc daune bonus de la proiectilele turnurilor.
- Înghețare: inamicul vor fi imobilizați.
- Bonus resurse: jucătorul primește un bonus de resurse cand inamicul este eliminat.
- Explozie: proiectilele turnurilor au un efect de explozie asupra inamicului și vecinilor săi.

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.

În majoritatea jocurilor, aceste sistem este prezent sub forma mecanismului de *buff* și *debuff*. Prin intermediul mecanismului de *buff*, un turn de apărare poate acorda beneficii suplimentare turnurilor vecine sau aliaților din jurul său. Acest lucru poate include creșterea puterii de atac, extinderea razei de acțiune, îmbunătățirea vitezei de tragere sau alte abilități speciale. Astfel, turnurile de apărare devin capabile să-și sprijine reciproc și să creeze sinergii puternice. Pe de altă parte, mecanismul de *debuff* implică capacitatea unui turn de apărare de a slăbi inamicii sau de a reduce eficacitatea atacurilor lor. De exemplu, un turn ar putea încetini inamicii din raza sa de acțiune, scăzându-le viteza de deplasare și facilitând astfel eliminarea lor de către alte turnuri. Aceasta poate deschide noi strategii de joc, permițând jucătorilor să controleze și să direcționeze mișcarea inamicilor într-un mod strategic.

Cea ce vom adauga nou la această idee este funcționalitatea de compunere. Această compunere crează noi jetoane cu un efect mai puternic, dar ca aceasta să se întâmple trebuie ca turnurile să fie plasate astfel încât inamicul să treacă prin raza lor de acțiune și să primească jetoanele respective.

Această tehnică de compunere este evidențiată în jocurile de tip *merge* – un gen de joc video care implică combinarea sau unirea elementelor sau obiectelor similare pentru a crea elemente noi și mai valoroase. Mecanica jocului se învârte în jurul potrivirii și unirii de obiecte identice

sau similare pe o tablă sau o grilă de joc, rezultând crearea de obiecte îmbunătățite sau avansate. Exemple de astfel de jocuri: *NecroMerge*, *2048*, *Top War*, *Rush Royal*, *Gold & Goblins* [12]. Aceste tipuri de jocuri pot fi cosiderate ca fiind „jocuri emergente" (*games of emergence*) – jocuri care au reguli relativ simple, dar multă variație [13].

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 de turnuri, fapt care crează redundanță. Trecând către acest sistem colaborativ, putem sesiza un nou beneficiu, și anume: jucătorul are mai multe oportunițăți să creeze noi energii și să experimenteze cu diferite combinații de turnuri având în vedere spațiul generos disponibil.

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 încăt să fie ușor de înțeles?
- Cum structurăm turnurile de apărare astfel încât să evidențiem acest sistem de jetoane de acțiune?
- Cum integrăm un sistem de progresie al jocului?
- Exista problema de conflict cu tematica aleasă? Dacă dezvoltam un joc cu tematică medievală, cum ar putea arată jetoanele de acțiune?
- Putem adauga un sistem multijucător? Care sunt provocările pe care le ridică acest sistem atunci când avem mai mult de un jucător uman?

Răspunsurile la aceste întrebări variază de la joc la joc, dar trebuie luate în considerare înainte de a începe dezvoltarea unui joc. Punerea în practică a unui astfel de sistem aduce cu sine o serie de noi întrebări și provocări. De multe ori nu vom avea răspunsuri la toate aceste întrebări, dar este important să avem o direcție clară în care să ne îndreptăm. Cea mai bună soluție e construirea unui mic prototip care să ne ajute să răspundem la aceste întrebări prin testare și experimentare înainte de a începe dezvoltarea jocului.

3.3. Compunerea jetoanelor de acțiune

În acest capitol vom explora ideea de compunere a jetoanelor de acțiune. Aceasta reprezentând și partea unică a acestei lucrări. Vom descrie cum ideile de mecanici de joc descrise în capitolul anterior (*buff*,*debuff*, *merge*) se pot combina și integra în jocul de *Tower Defense* sub forma unui sistem de jetoane de acțiune.

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 (Figura 2). De exemplu, putem avea următoarele tipuri de jetoane compuse:

- Jeton de înghețare: inamicii vor fi înghețați (viteză de mișcare 0) atâta timp cât 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 (Figura 3).

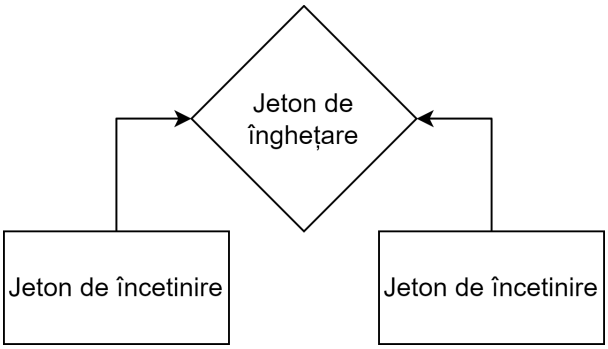


Figura 2: Formarea unui jeton compus.

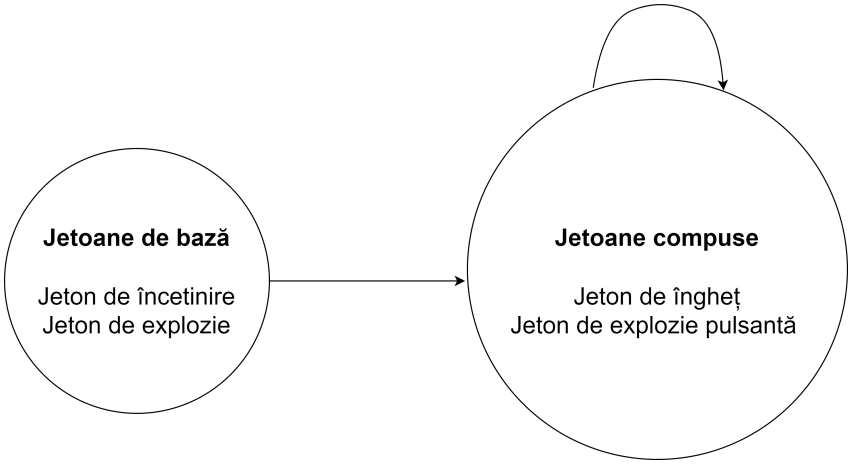


Figura 3: Extinderea jetoanelor de bază în cele compuse.

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* care se bazează pe extensii pentru a varia experiența de joc. Aceste extensii aduc noi tipuri de cărți care pot fi combinate cu cele existente pentru a crea noi strategii de joc.

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 4). 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 precum 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.
- Compatibilitate: dacă dezvoltăm o extensie de joc care aduce noi tipuri de jetoane, acestea trebuie să fie compatibile cu cele existente.
- Monotonie: mereu sunt alese aceleași jetoanre întrucât celelalte nu sunt suficient de puternice, eficiente sau nu se potrivesc cu strategia de joc.
- Lipsa unei experiențe graduale: curba de dificultate este destul de abruptă întrucât noțiunile despre teoria jetoanelor de acțiune nu este bine explicată.
- Balansarea și ajustarea constantă: Pentru a menține echilibrul și a asigura o experiență de joc corectă, este important să se efectueze ajustări și balansări periodice. Acest proces este destul de dificil și consumă mult timp.
- Moduri de joc alternative: uneori de dorim să avem și mici provocări care să ne testeze abilitățile de joc. Acestea pot fi introduse sub forma unor moduri de joc alternative care să ofere o experiență de joc diferită. Ca acest lucru să fie posibil trebuie să reevaluăm din nou sistemul implementat.

Toate aceste probleme trebuie abordate înainte de a începe dezvoltarea unui joc. O planificare atentă și o analiză profundată a acestor probleme ne poate ajuta să evităm multe probleme pe

parcursul dezvoltării jocului. Rolul unei prototip care să testeze și să verifice aceste probleme este esențial în dezvoltarea unui joc de calitate.

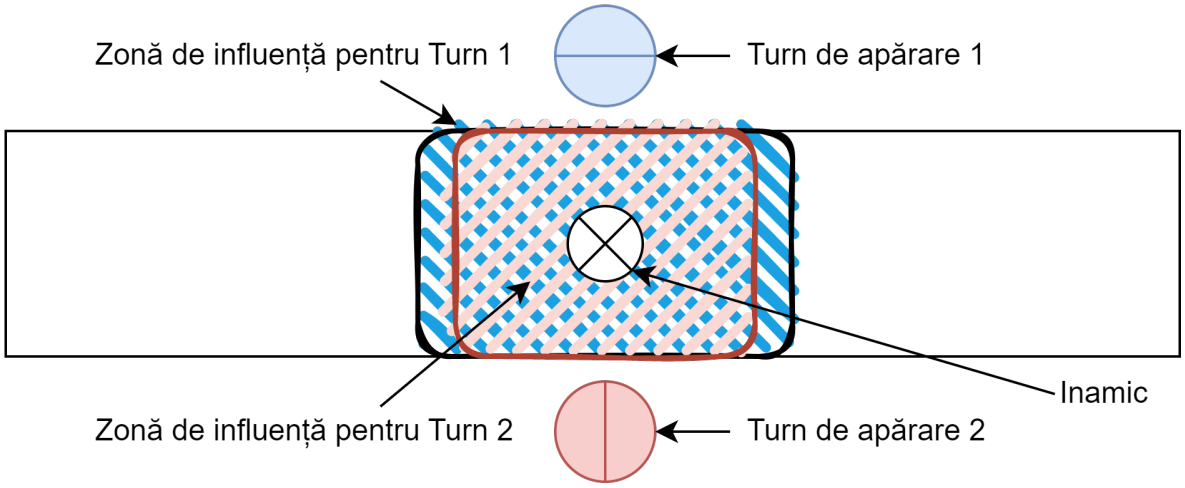


Figura 4: Conflict la compunere.

3.4. 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 provoacă 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 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

Principalele caracteristici ale unui turn:

- Puterea de atac: reprezintă pagubele pe care le poate produce un turn de apărare.

- Rata de atac: reprezintă cât de repede poate ataca un turn de apărare.
- Raza de acțiune: reprezintă distanța maximă până la care un turn de apărare poate ataca.
- Modul de țintire: atacă o singură țintă sau mai multe.

Este important ca aceste valori să fie echilibrate și in baza lor să facem diferite variații. De asemenea, putem schița și tipurile de îmbunătățiri pe care le poate avea un turn de apărare:

- Creșterea puterii de atac: crește puterea de atac a turnului.
- Creșterea razei de acțiune: crește raza de acțiune a turnului.
- Creșterea ratei de atac: crește rata de atac a turnului.
- Creșterea numărului de ținte: crește numărul de inamicii care pot fi atacați simultan.

De menționat este faptul că nu toate turnurile pot avea îmbunătățiri pentru toate caracteristicile în cele mai multe cazuri, lucru care depinde și de design-ul jocului per total. Dacă avem o hartă mare, am putea mării raza de acțiune a turnurilor, dar dacă avem o hartă mică, trebuie să fim consecvenți întrucât am putea crea un dezechilibru în joc din cauză ca spațiul este foarte limitat.

Iar în privința îmbunătățirilor, acestea pot fi progresive sau unice. Cele progresive sunt îmbunătățiri care pot fi aplicate de mai multe ori (puterea turnurilor crește puterea gradual de-a lungul sesiunii de joc), iar cele unice sunt îmbunătățiri care pot fi aplicate o singură dată (acestea includ avantaje semnificative care schimbă radical valoarea turnului).

Echilibrarea turnurilor este un proces meticulos care necesită multă atenție, iar dacă punem și în considerare îmbunătățirile și jetoanele de acțiune, acest proces devine și mai dificil. Stabilirea unor valori inițiale pentru caracteristicile turnurilor este un prim pas, iar apoi putem face ajustări pe parcursul dezvoltării jocului. O functionalitate care ne poate ajuta în acest sens este cea de a înregistra sesiunile de joc și de a analiza datele colectate. Aceste date ne pot ajuta să identificăm problemele și să le rezolvăm.

3.5. 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

De asemenea, într-o rundă putem mai multe tipuri de inamicii. De exemplul, valul poate să înceapă cu cei mai slabi și să termine cu cei mai puternici. Acest lucru ne permite să creăm o curba de dificultate care să crească pe măsură ce jocul avansează. Alte variante sunt:

- Inamici cu abilități speciale: Un inamic ar putea avea capacitatea de a se teleporta prin turnurile tale sau de a se vindeca pe parcursul luptei. Acești inamici speciali vor necesita strategii diferite pentru a fi învinși.

- Inamici care acționează în grupuri sau formații: Aceștia pot avea tactici de atac diferite, cum ar fi acoperirea unor inamici mai slabi sau protejarea unui lider puternic. Astfel, jucătorii vor trebui să găsească modalități eficiente de a rupe aceste formații pentru a-i învinge.
- Inamici cu vulnerabilități temporare: un inamic ar putea fi invincibil în mod normal, dar să aibă un moment vulnerabil în timpul rundei în care poate fi atacat. Acest lucru va implica planificare și sincronizare din partea jucătorilor pentru a profita de momentul oportun.
- Inamici finali: caractere speciale care apar la sfârșitul unei runde și care sunt mult mai puternici decât inamicii obișnuiți.

Un exemplu de astfel de joc care dispune de o vastă varietate de inamicii este *Kingdom Rush* [14] – acesta poate fi considerat ca find un punct de referință în ceea ce privește design-ul inamicilor.

Una din limitele sistemului de jetoane este design inamicilor. Având un număr mare de inamicii unici putem crea diverse jetoane care să acopere o gamă largă de posibilități. Dar trebuie sa avem grijă cu reduntanța, întrucât putem crea jetoane care să facă același lucru într-un mod puțin diferit.

Un mod popular de diversificare este introducerea unui sistem de armură care să contracareze anumite tipuri de atac. Astfel, putem avea inamicii cu armură de foc, armură de gheață, armură de magie, etc. Aceste tipuri de armură pot fi contracarate de jetoanele de acțiune care oferă un bonus de atac pentru un anumit tip de armură. Se poate observa cum putea multe tipuri de jetoane care se aseamănă la scop. Cu puțină creativitate putem crea niște turnuri care se folosesc în mod unic de aceste jetoane pentru a crea o nouă experiență de joc.

O altă idee creativă este crearea de inamici care reacționează la jetoanele de acțiune. Dacă un inamic primește un jeton de atac bonus, își crește mișcare ca să poată că să scape mai repede din raza inamicilor. Dacă este înghețat din cauza jetonului de îngheț, își poate crește nivelul de armură pentru a reduce pagubele primite. Acestă nouă perspectivă poate fi folosită pentru a crea inamicii care să fie mai dinamici și mai imprevizibili. O idee care merită explorată.

3.6. 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 următoarele costuri pentru turnurile de apărare din Tabelul 3 și 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 unul 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 această suma el și-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.

Un joc care prezintă o sistem simplu de economie și în jurul căruia este contruită toate mecanica de joc este *Kingdom: Clasic* [15] – în care singura sarcină a jucătorului este distribuirea resurselor între clădirile de apărare și cele de producție. Chiar și cu această sarcină singulară, jocul oferă o experiență de joc foarte bună. Iar în versiunea de *Kingdom: Two Crowns* se adaugă și partea de colaborare cu un alt jucător – fapt ce arată versatilitatea design-ului de joc [16].

4. 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.

Un alt aspect important este alegerea modului în care este structurată logica jocului. Și anume dacă ne dorim ca tot jocul să fie un automat finit sau să folosim un sistem dinamic de evenimente.

Un automat finit este un model matematic care descrie comportamentul unui sistem cu un număr finit de stări și tranziții între aceste stări. În cadrul jocului, aceasta ar însemna că există o secvență predefinită de evenimente și acțiuni care se întâmplă într-un anumit ordine și care determină progresul jocului. Precum în șah, unde avem o secvență de mutări care trebuie urmate pentru a ajunge la un rezultat final. Partea avantajoasă e că putem analiza sesiunea de joc care este mereu reproductibilă – lucru care poate face mai ușoară analiza sesiunilor de joc și echilibrarea jocului. Partea dezavantajoasă este că putem folosii functionalități care nu sunt disponibile într-un automat finit, precum: evenimente aleatorii, evenimente care se întâmplă în paralel.

Într-un sistem dinamic de evenimente, jocul poate răspunde în timp real la acțiunile jucătorului și la alte factori variabili. Evenimentele pot fi declanșate de acțiunile jucătorului sau de anumite condiții în joc și pot avea efecte imediate sau pe termen lung. Evenimente aleatorii care apar în mod imprevizibil pot avea impact asupra jocului, cum ar fi o furtună care afectează resursele sau apariția unui bonus surpriză într-o anumită zonă a hărții. Aceste evenimente adaugă o doză de surpriză și incertitudine în joc, dar pot fi mai dificile de echilibrat și de analizat întru-cât sesiunile de joc nu sunt mereu reproductibile cu sesiunea de referință.

Majoritatea jocurile de *Tower Defense* folosesc un sistem dinamic de evenimente având moșternirea din jocurile de strategie în timp real. În această lucrare ne vom concentra mai mult pe partea de automat finit întrucât este mai ușor de analizat fapt care ne ajută în evidențierea sistemului de jetoane de acțiune și modului cum acesta poate fi folosit într-un joc de *Tower Defense* – mecanica de joc devine discretă [13]. De asemenea, nu excludem faptul se pot folosii ambele sistem în mod hibrid, întrucât acestea nu sunt incompatibile.

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 [17]. 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` [18] care ne permite schimbarea de module și sisteme 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 a structura codul astfel încât să fie ușor de înțeles și de extins. De exemplu, în cazul nostru, vom folosii *design pattern*-ul *Observer* [19] pentru a implementa sistemul de evenimente. Inamicii, proiectilele și turnurile vor fi subiecte, iar sistemele centrale vor fi observatori.

Alte astfel de *design patterns* [19] [20] 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 tehnicile enumerate anterior ne ajută să păstrăm un cod curat și ușor de înțeles. Însă acestea nu rezolvă problema performanței.

Performanța depinde foarte mult de modul cum gestionăm memoria și procesorul. Mereu vom dori 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) [21]. 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* [22].

La o primă vedere acest sistem pare a fi foarte complex, dar trebuie să avem în vedere că sistemul de jetoane de acțiune este un sistem cu multe entități și reguli de transformare. Pentru joc de anvergură mică sau medie, nu ar fi necesară folosirea acestui sistem. Dar pentru jocuri de buget mare, acest sistem este foarte util întrucât ne permite să salvam resurse care ar putea fi folosite pentru animații și efecte speciale, acestea fiind mari consumatoare de resurse.

Pentru aplicația de demonstrație vom folosii tehnologiile WEB pentru a crea un document interactivi care să prezinte sistemul de jetoane de acțiune. Pentru partea de implementare vom folosii limbajul de programare Typescript și biblioteca SolidJs pentru implementarea interfeței [23]. Aceasta ne permite să creăm vizualizări complexe și interactive într-un mod foarte ușor.

4.1. Primele minute de joc

Inainte de a intra în detalii despre implementarea sistemului de jetoane de acțiune, vom descrie cum ar arăta 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.

Interfața de magazin ne va arată următoarele opțiuni:

- Turnuri de apărare disponibile pentru achiziționare, inclusiv costul și atributele acestora, în special tipurile de jetoane de acțiune pe care le pot crea sau utiliza.
- Îmbunătățiri pentru turnuri și jetoane împreună cu costul acestora și trasătura pe care o îmbunătățesc.

Magazinul este foarte important întrucât jucătorii își vor petrece multe timp aici pentru a se decide ce turnuri de apărare să cumpere și ce îmbunătățiri să facă. Este important ca această interfață să fie cât mai intuitivă și ușor de folosit.

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.

Deoarece folosim un automat finit în implementarea sistemului de joc din cadrul aplicației putem avea opțiunea de a relua sesiunea de joc. Acest lucru ne permite să analizăm modul cum am jucat și să încercăm să găsim o strategie mai bună. De asemenea, putem să relua runda și să încercăm o nouă tactică, să vedem dacă aceasta este mai bună sau nu.

4.2. Harta de joc

Harta de joc este o zonă în care se desfășoară acțiunea jocului. În jocurile de *Tower Defense* harta este împărțită în doua tipuri:

- Hartă liberă: nu există un trase prestabilit, iar prin plasarea turnurilor de apărare putem influența traseul inamicilor întrucât aceștia aleg de fiecare dată cel mai scurt drum până la obiectiv.
- Hartă cu trase prestabilit: există un traseu prestabilit pe care inamicii îl vor urma. În acest caz, plasarea turnurilor de apărare nu influențează traseul inamicilor, dar forma drumului influențează puternic eficiența turnurilor de apărare.

În multe jocuri harta cu traseu prestabilit este preferată întrucât permite posibilitatea de a creea mai multe nivele de joc întrucât mici diferențe în traseu pot duce la strategii diferite. Un alt avantaj este integrarea mai mult rute de atac, ceea ce poate duce la o experiență de joc mai variată.

Harta liberă are un avantaj prin faptul ca partea de plasare a turnurilor joacă un rol și mai important în special ca ele vor dicta treaseul inamicilor. Un mic impendiment la acest tip de design este faptul că jucătorul va incerca mereu să obțina forma optima a treaseului pentru inamicii, iar această formă nu are multe variații cea ce face ca jucătorul să construiască aceeași formă de fiecare dată. Acest impediment poate fi rezolvat prin adăugarea de elemente de decor care să dicteze o anumită formă a traseului.

Unele jocuri, precum *Kingdom Rush* [24], folosesc o combinație între cele două tipuri de hărți. Acestea au un traseu prestabilit, dar în același timp au și elemente de decor care dispar în timpul jocului pentru a crea rute de atac adiționale astfel mărind dificultatea nivelului.

În aplicație vom folosii o hartă cu trase prestabilit întrucât este mai ușor de implementat și ne permite să ne concentrăm pe partea de jetoane de acțiune. Harta va fi o matrice de căsuțe de dimensiune 8 pe 12 (Figura 5). Căsuța verde va reprezenta punctul de start de unde va începe valul de inamicii, iar căsuța roșie va fi finalul traseului. Căsuța galbenă va fi căsuța curent selectată de către jucător. Dacă pe căsuța selectată se află un turn de apărare, atunci raza ei de acțiune va fi vizibilă, iar dacă se află un inamic, atunci informațiile despre acesta vor fi vizibile în panoul din dreapta.

Traseul este dat de căsuțele de culoarea asemănătoare turcoazului. Inamicii vor urma acest traseu de la punctul de start până la obiectiv. În cazul în care un inamic ajunge la obiectiv, viața obiectivului va scădea. Dacă viața obiectivului ajunge la 0, jocul se va termina. Turnurile de apărare pot fi construire doar în afara traseului, și anume pe căsuțele de culoarea asemănătoare a verdelui închis.

Mai mult de atât, forma traseului va influența puternic eficiența turnurilor de apărare. De exemplu, dacă traseul este foarte îndreptat și fără obstacole, turnurile cu raza lungă de acțiune, cum ar fi cele de tip artilerie, vor fi extrem de eficiente. Acestea pot lovi țintele de la distanțe mari și pot cauza daune semnificative înainte ca inamicii să se apropie prea mult. Turnurile care încetinesc inamicii pot fi foarte valoroase, deoarece permit altor turnuri să lanseze mai multe atacuri înainte ca inamicii să scape din raza de acțiune.

Pe de altă parte, pe traseele sinuoase sau cu multe curbe, turnurile de scurtă rază de acțiune care pot ataca mai multe ținte simultan vor fi mai eficiente. De asemenea, pe aceste trasee, amplasarea strategică a turnurilor care pot ataca în mai multe direcții va fi esențială. De exemplu, turnuri care pot lansa flăcări sau explozii preferă zone cu curbe strânse, unde inamicii tind să se adune.

În plus, amplasarea turnurilor în apropierea locurilor în care traseul se intersectează cu sinele are un efect major, deoarece acestea pot ataca inamicii care trec de mai multe ori. Acest lucru poate fi valabil și pentru traseele cu mai multe ramificații, unde turnurile cu rază mare de acțiune pot acoperi mai multe căi simultan.

În Figura 5 putem observa că traseul are o formă sinuoasă la început și spre sfârșit, ideal pentru turnuri cu rază scurtă și atac rapid. De asemenea, putem observa că există două locuri în care traseul este lung și drept, tocmai bun pentru turnuri cu rază lunga.

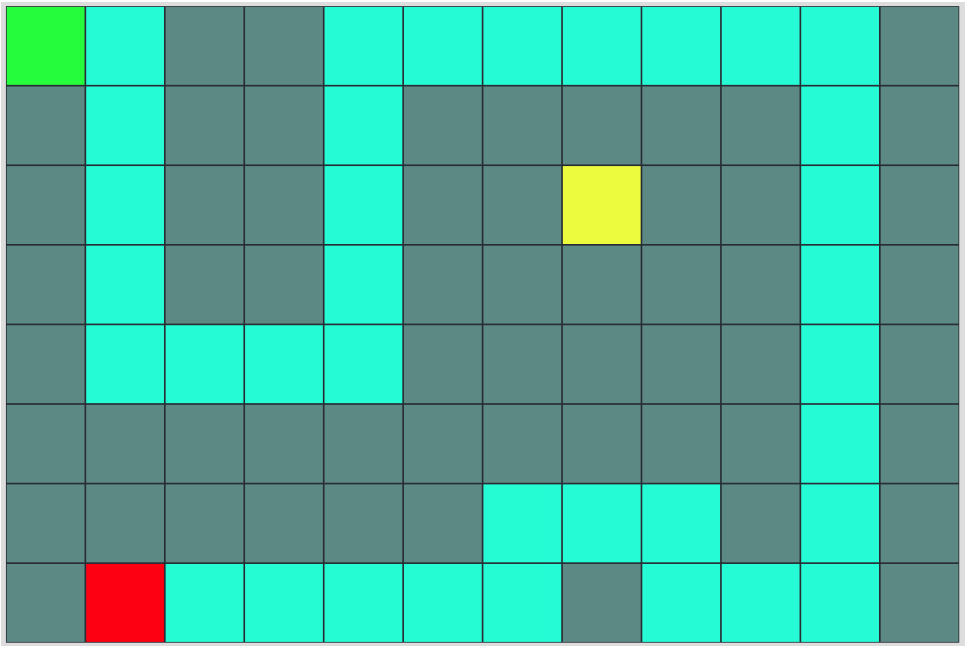


Figura 5: Reprezentarea hărții de joc.

4.3. 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 timpul 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 timpul 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 Typescript[25] 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:

```
type Token = {
  effect_type: EffectType,
  effect_value: number,
  max_rank: number,
  current_rank: number,
  duration: number,
  current_duration: number,
  creation_conditions: Array<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:

```
type Condition = {
  effect_type: EffectType,
  min_rank: number
}
```

În jocul din cadrul aplicației fiecare jeton de acțiune este reprezentat de către o iconiță unică care este afișată de către inamic. Aceasta este o reprezentare vizuală a jetonului de acțiune care ne ajută să vedem ce jetoane de acțiune are inamicul și ce rang au acestea (Figura 6).

Figura 6: Reprezentarea jetoanelor de acțiune pe inamic.

În magazin avem posibilitatea să cumpărăm îmbunătățiri pentru jetoane de acțiune (Figura 7). Acestea includ îmbunătățiri pentru valoarea efectului sau duratei de timp. Fiecare variază în funcție de preț și de efectul pe care îl îmbunătățește.

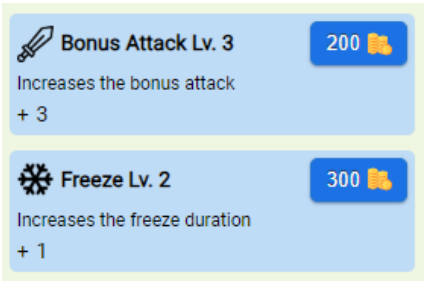


Figura 7: Reprezentarea îmbunătățirilor pentru jetoane de acțiune în magazin.

4.4. 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 proiectilului: 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.

Întrucât în implementarea jocului nu avem nevoie de proiectile, acestea tot vor fi menționate în descrierea sistemului de turnuri de apărare, deoarece sunt o componentă care apar în majoritatea jocurilor de *Tower Defense*.

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.

Tipurile de date ale structurii pentru un turn de apărare activ și proiectil sunt următoarele:

```
type Projectile = {
  speed: number,
  damage: number,
  life_time: number,
  current_life_time: number,
}
```

```
type ActiveTower = {
  projectile: Projectile,
  attack_rate: number,
  attack_rate_timer: number,
  range: number,
}
```

Pentru turnurile pasive avem următoarea structură:

```
type PassiveTower = {
  token: Token,
  range: number,
  token_rate: number,
  token_rate_timer: number,
}
```

Acest sistem este simplu de implementat, cea mai complicată partea find partea de țintire a turnurile active. In mod ideal, am dori 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. Și rezolvarea acestei problemă necesită să facem o simulare a jocului în timp pentru a determina viața inamicului la momentul dorit.

Dacă nu folosim proiectile și atacul din partea turnului este instant, atunci devine mult mai ușor să stabilim viața inamicului la momentul în turnul este pregătit să tragă prin faptul ca putem verifica instant viața curenta a inamicului. Tot ce trebuie să stabilim e un sistem de priorități în cadrul turnurilor de apărare. Putem folosii o metodă de tip *greedy* [26] (aceasta implică selectarea, la fiecare pas, a opțiunii care pare cea mai bună la acel moment, fără a lua în considerare consecințele pe termen lung sau efectele pe care decizia respectivă le-ar putea avea asupra soluției finale) prin care stabilim care sunt turnurile necesare pentru a elimina inamicul folosind un număr minim de turnuri. Astfel, putem stabili o ordine de tragere a turnurilor în funcție de prioritatea lor.

În aplicație tunurile vor fi reprezentate de imaginii sugestive aflate pe harta de joc (Figura 8). În casuța turnului, pe colțul din stânga jos este un simbol care arată dacă turnul este pregătit să tragă sau nu. Dacă nu, va arată timpul rămas până la următorul atac. De asemenea, la selectarea turnului de apărare, raza de atac va fi vizibilă prin schimbarea culorii din căsuțele învecinate turnului (Figura 9).

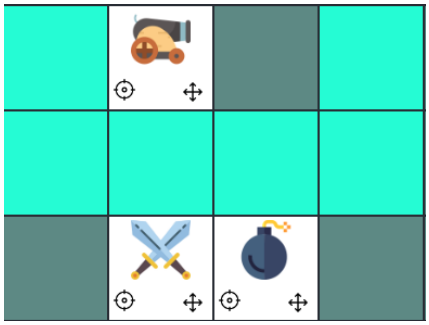


Figura 8: Reprezentarea turnurilor de apărare pe harta de joc.

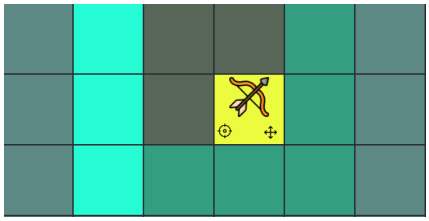


Figura 9: Reprezentarea razei de atac a turnurlui de apărare.

În magazin avem reprezentarea turnurilor (Figura 10) și îmbunătățirilor disponibile pentru achiziționare (Figura 11). Fiecare reprezentare are un număr care arată costul acestora, iconița indicând imaginea turnului care este vizat și o descriere a turnului sau îmbunătățirii. Pentru turnuri avem iconițe care indică puterea de atac, raza de acțiune, rată de atac și modul de atac (dacă este cazul - exemplu: *Bomb Tower*), pe langă acestea avem și tipurile de jetoane de acțiune pe care le poate crea sau utiliza. Pentru îmbunătățiri avem iconițe care indică ce trasătură îmbunătățește și cu cât.

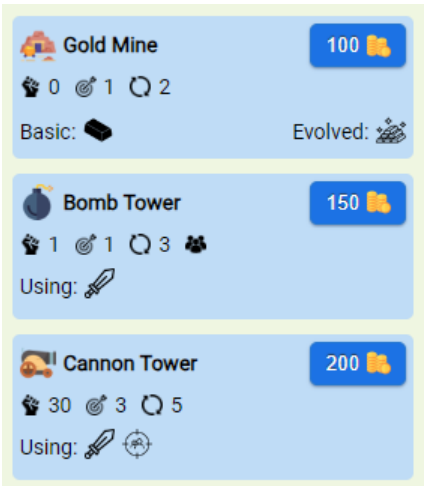


Figura 10: Reprezentarea turnurile de apărare în magazin.

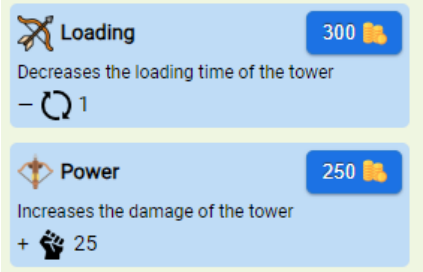


Figura 11: Reprezentarea îmbunătățirilor în magazin.

4.5. 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 jocurile care se combină cu *Role Play Game* (RPG), această parte devine fundamentală pentru joc. Un astfel de joc este *Arknight*[27] 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.

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. Exemple de structuri pentru tipul de resursă și recompensă sunt următoarele:

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

```
type Reward = {
  resource_type: ResourceType,
  amount: number,
}
```

Știind câți inamicii avem într-un val și care este valoarea lor totală, 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 prezente 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 extragă 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.

Jocul din cadrul aplicației prezintă o variantă simplificată cu o singură resursă numită *monedă*. Prin eliminarea inamicilor, jucătorul primește o recompensă sub formă de monezi. Monezile poate fi folosită pentru a achiziționa turnuri de apărare și îmbunătățiri pentru acestea prin intermediul magazinului. Figura 12 este reprezentarea vizuală a magazinului – în colțul din dreapta sus avem numărul de monezi pe care le deținem, iar în partea de jos avem turnurile de apărare și îmbunătățirile disponibile pentru achiziționare și fiecare având un cost de achiziție.

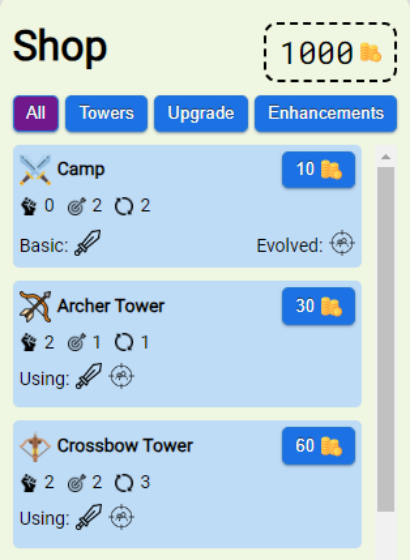


Figura 12: Reprezentarea vizuală a unui magazin pentru achiziționarea de turnuri de apărare și îmbunătățiri.

Utilitatea turnurilor este reflectată și in prețul lor. In Figura 12, un turn de arcași (*Archer Tower*) are o rază mai mică de atac decât un turn de arbaletiști (*Crossbow Tower*), cea cel face mai puțin ideal întrucât nu are posibilitatea să atace inamicii de mai multe în părțile sinoase ale traseului. De aici și ideea de a avea un preț mai mic decât cel al turnului de arbaletiști.

Un alt truc pe care îl mai putem face să variem alegerea de turnuri este prin modifcare dinamică a prețului. Un exemplu ar fi dacă după fiecare achiziție prețul ar crește de 5 ori pe tipul respectiv de turn. Însă trebuie să avem grijă cu această idee întrucât se poate crea un dezechilibru prin faptul ca unele turnuri nu vor fi folosite niciodată din cauza prețului lor mare în raport cu valoarea adusă.

4.6. 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:

```
type Enemy = {
  health: number,
  speed: number,
  tokens: Array<Token>,
  reward: Reward,
  traits: Array<string>,
}
```

Unde *traits* reprezintă abilitățile/trăsăturile speciale pe care le poate avea inamicul. Acestea pot fi reprezentate prin simple șiruri de caractere, precum: ”rezistență la foc” (*fire resistance*), ”regenerare” (*regeneration*), etc.

Un val de inamicii este o listă de inamicii care trebuie să ajungă la obiectivul care trebuie protejat de-a lungul unui runde joc. Această listă variază în funcție de progresului jucătorului în joc. De exemplu, la începutul sesiunii de joc avem un val de 10 inamicii, iar la finalul ei avem

un val de 100 inamicii. Tipurile de inamicii variază de asemenea de val la val. De exemplu, la începutul jocului avem 10 inamicii normali, iar la finalul jocului avem 20 inamicii rezistenți, 30 rapizi și 50 normali.

Un val (*wave*) poate fi descris de următoarea structură:

```
type EnemyNum = {
  enemy_type: EnemyType,
  amount: number,
}
```

```
type Wave = {
  enemies: Array<EnemyNum>,
}
```

O chestie importantă este intervalul de generare al inamicilor în cadrul valului. Dacă intervalul este mic se poate crea un grup de inamicii care să ajungă la obiectiv înainte ca turnurile de apărare să poată să-i elimine. Dacă intervalul este mare, atunci turnurile de apărare pot să elimine inamicii fără prea mult efort sau strategie din parte jucătorului. Deci, trebuie să găsim un echilibru între aceste două extreme.

Jocuri precum *Kingdom Rush* variază acest interval în cadrul rundei de joc pentru a crea grupuri de inamicii mai mici sau mai mari pentru a oferi o provocare jucătorului prin prisma faptului că inamicii mai rezistenți pot distrage turnurile de apărare, iar cei mai rapizi pot ajunge la obiectiv înainte ca turnurile de apărare să-i elimine.

În Figura 13 putem observa implementarea valului de inamicii din cadrul aplicației. Punctul de start este pătratul verde, iar punctul de final este pătratul roșu. Inamicii sunt reprezentați de imagini care prezintă niște creaturi ficționale de culoare roșie, iar în colțul din dreapta sus este un număr care reprezintă punctele de viață. Inamicii se deplasează de la punctul de start la punctul de final. Cei care ajung la punctul de final sunt retrași de pe hartă, iar jucătorul primește o penalizare. Iar cei care sunt eliminați de către turnurile de apărare (Figura 14) oferă o recompensă jucătorului.

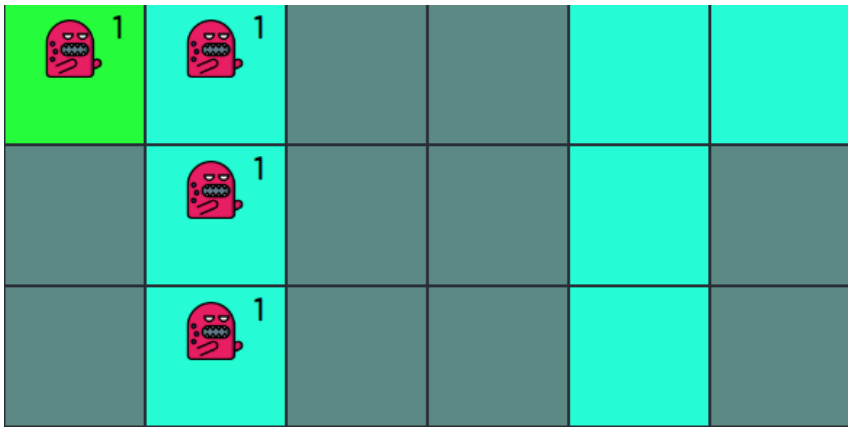


Figura 13: Reprezentarea vizuală a unui val de inamici.

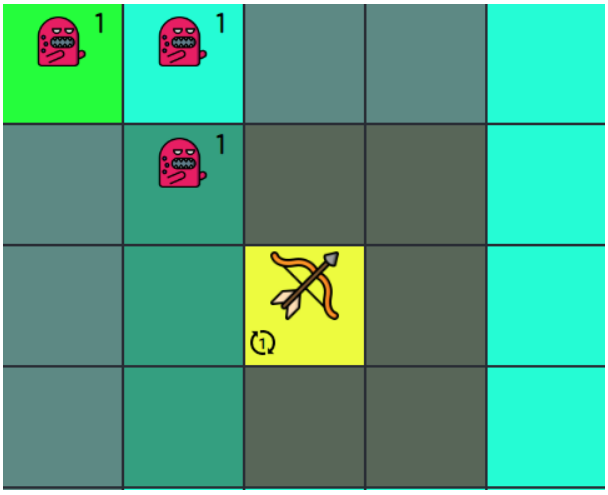


Figura 14: Inamic în raza de acțiune a unui turn de apărare.

Cum inamicii se află într-o hartă de joc de tip matrice și cu o viteză de mișcare egală cu o celulă, noțiunea de încetinire nu are prea mult sens întrucât nu avem diviziunii în cadrul celulei. De aceea, am ales să implementăm doar jetoanele de înghețare care îngheață inamicul pentru o perioadă de timp. Inamicii înghețați nu se pot mișca, iar cei din spatele lor nu pot să treacă prin ei – fapt care creează un ambuteiaj. Față de o implementare clasică, această alegere de mișcare a inamicilor într-o matrice cu o celulă pare să fie foarte limitată în privința designului de inamicii. Dar nu este un impediment major, se pot găsi soluții creative pentru a trece peste această limitare.

4.7. 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 informații pe care elementele vizuale din interfața de utilizator trebuie să le afișeze:

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

În Figura 15 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 [28], [29]:

- 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 16 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 17 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 18 este schița pentru un astfel de element vizual.

În aplicație, interfața a fost creată folosind tehnologii WEB precum *HTML*, *CSS* și *Javascript*. Acestea sunt tehnologii care sunt folosite pentru a crea interfețe de utilizator pentru aplicații web. Pe langă acestea, folosim și biblioteca SolidJs care ne ajută să creăm interfețe de utilizator într-un ritm rapid și ușor. Aceasta se bazează pe conceptul de *reactive programming* [30] care ne ajută să creăm documente WEB care se actualizează automat atunci când starea internă a aplicației se schimbă în urma interacțiunilor cu utilizatorul. Aceasta este o bibliotecă care este inspirată de biblioteca *React* [31] care este foarte populară în comunitatea de dezvoltare a aplicațiilor WEB.

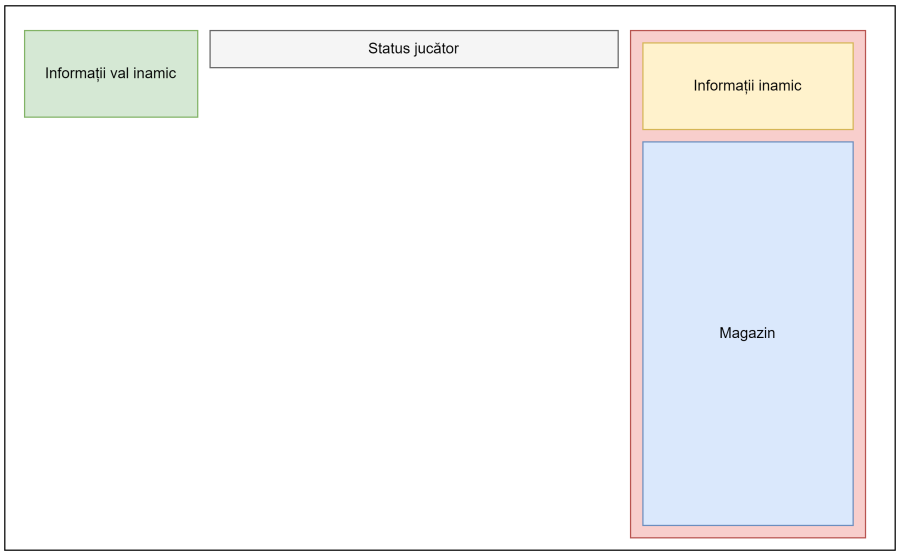


Figura 15: Schiță pentru interfața de utilizator.

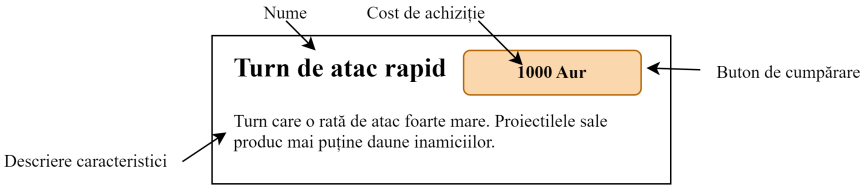


Figura 16: Schiță pentru un element vizual al unui turn activ din magazin.

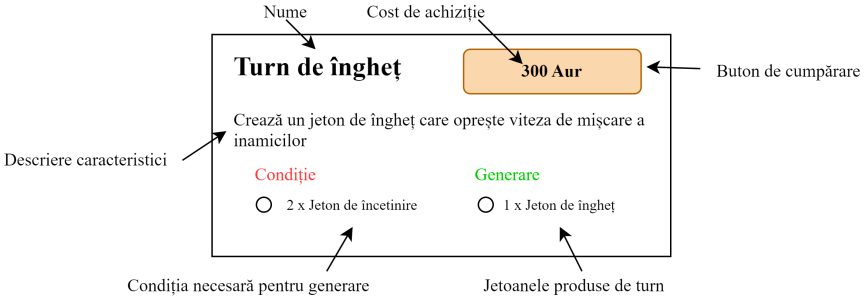


Figura 17: Schiță pentru un element vizual al unui turn pasiv din magazin.

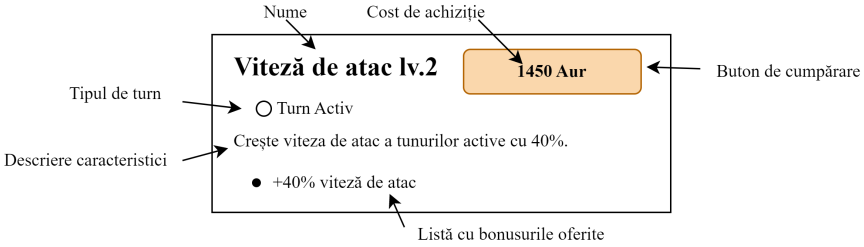


Figura 18: Schiță pentru un element vizual al unui îmbunătățiri din magazin.

Figura 19 este o reprezentare vizuală a interfeței de utilizator din cadrul aplicației dezvoltate. În partea din stânga avem magazinul, iar în partea din dreapta avem informații despre câți inamicii au rămas și numărul valului curent. În partea centrală se află hart de joc reprezentată sub forma unei matrici de căsuțe.

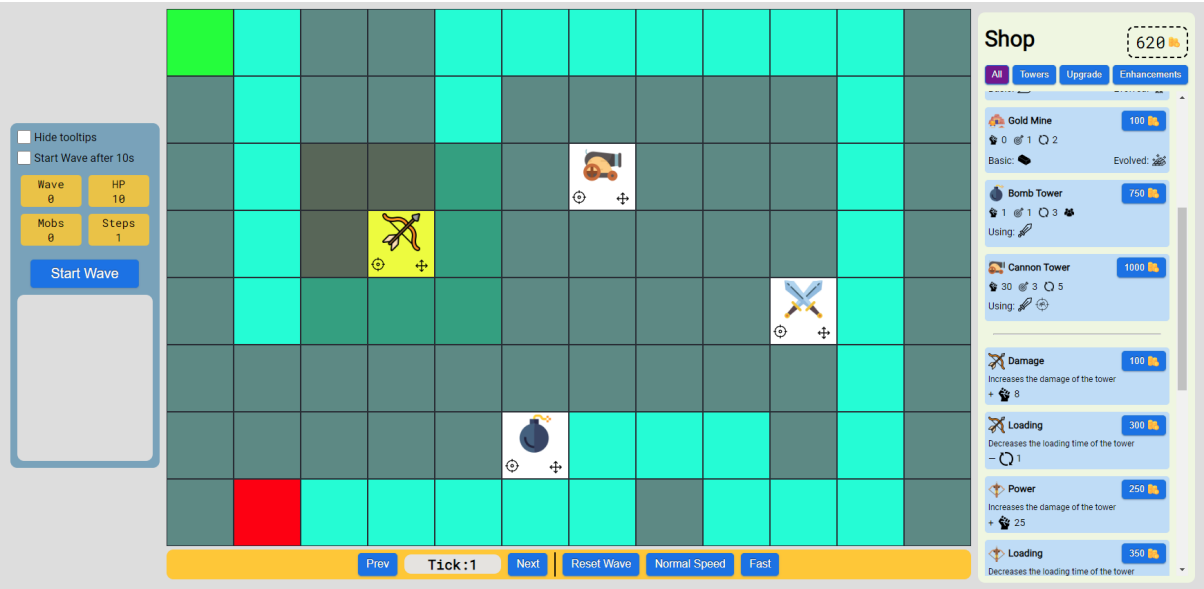


Figura 19: Vedere de ansamblu al interfeței de utilizator.

Căsuța de culoare verde indică punctul de start al valului de inamici, iar căsuța de culoare roșie este obiectivul la care aceștie trebuie să ajungă. Căsuțele de culoare asemănatoare turcuazului reprezintă calea pe care inamicii trebuie să meargă. Căsuțele de culoarea asemănătoare verdului închis reprezintă zonele în care putem construi turnuri de apărare.

În partea de jos avem controale pentru sesiunea de joc. Pe partea stângă avem controlul pentru redarea pașilor pe care îl putem folosii să analizăm mai în detaliu modul cum a decurs runda de joc. Pe partea dreaptă avem controlul pentru pauză care ne ajută să oprim jocul și opțiunile de viteză în caz că dorim ca desfășurarea rundei să fie mai rapidă.

În panoul din dreaptă avem mici opțiuni precum *Hide Tooltip* – care activează sau dezactivează mesajele ajutătoare (Figura 22) – și *Start wave after 10s* care pornește automat următorul val de inamicii după 10 secunde de la terminarea valului precedent.

În Figura 21 avem informații despre statusul jocului, cu următoarele elemente vizuale:

- numărul valului de inamici curent (*Wave*);
- numărul de inamicii rămași (incluzând și pe cei care urmează să fie generați) în valul curent (*Mobs*).
- numărul de puncte de viață rămase pentru obiectivul care trebuie protejat (*HP*).
- numărul total de pași de joc care au fost executați (*Steps*).

Figura 20: Menu de control al pasului de joc.

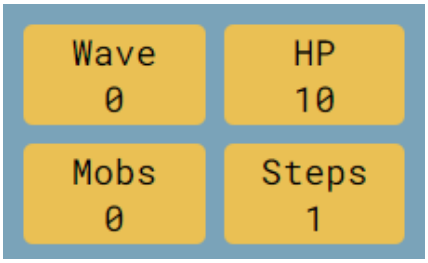


Figura 21: Statutul jocului.

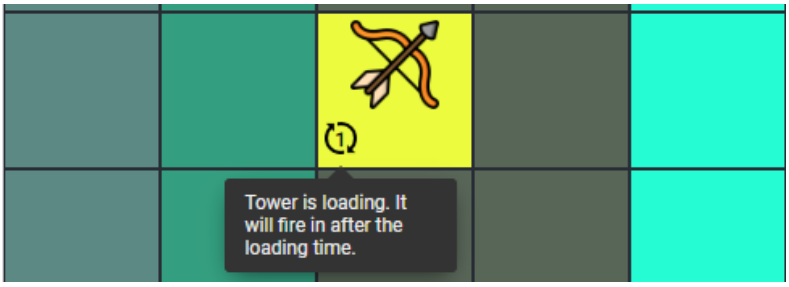
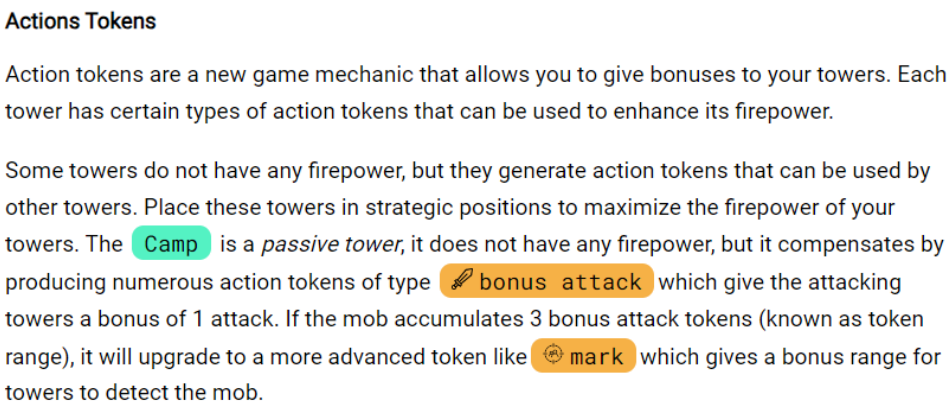


Figura 22: Mesaj ajutător care explică un elemente din joc.

Un avantaj pe care îl prezintă tehnologiile web este usurința cu care putem crea documente menite să prezinte informația într-un mod clar și concis. Documentația despre mecanica de joc este inclusă în aplicație și poate fi accesată cu usurință chiar și în timpul sesiunii de joc (Figura 23).

Un singur dezavantaj la această tehnologie este faptul că nu este ușor să implementăm animații complexe precum într-un motor de joc dedicat pentru crearea de jocuri. Cu toate acestea, această tehnologie este o soluție elegantă pentru crearea de prototipuri care pot fi împărtășite cu usurință în cadrul comunității.



Actions Tokens

Action tokens are a new game mechanic that allows you to give bonuses to your towers. Each tower has certain types of action tokens that can be used to enhance its firepower.

Some towers do not have any firepower, but they generate action tokens that can be used by other towers. Place these towers in strategic positions to maximize the firepower of your towers. The `Camp` is a *passive tower*, it does not have any firepower, but it compensates by producing numerous action tokens of type `bonus attack` which give the attacking towers a bonus of 1 attack. If the mob accumulates 3 bonus attack tokens (known as token range), it will upgrade to a more advanced token like `mark` which gives a bonus range for towers to detect the mob.

Figura 23: Instrucțiuni de joc despre mecanismul de jetoane de acțiune.

4.8. Unelte pentru analiza jocului

Datorită faptului ca mediul interactiv din cadrul aplicației este modelat ca un automat finit, putem analiza jocul prin schimbările de stare al acestuia. Fiecare turn cumpărat sau mutat, fiecare îmbunătățire achiziționată este înregistrată sub forma unui eveniment. Aceste evenimente pot fi folosite pentru a analiza jocul și pentru a crea statistici despre acesta.

Precum în șah, jocul este descris de o secvență de mutări. Mutările sunt codate folosind un sistem de notații algebrice [32] care ne ajută să descriem o mutare într-un mod concis. De exemplu, mutarea de start *Nf3* descrie mutarea unui cal de pe poziția *g1* pe poziția *f3*. Mutarea *Bb5* descrie mutarea unui nebun de pe poziția *c1* pe poziția *b5* (fiind prima mutare a piesei).

Mutările sunt înregistrate într-un jurnal de joc (*game log*) care este folosit pentru a analiza jocul. Acest jurnal poate fi interpretat ca o istorie a stărilor prin care a trecut jocul. Utilizând acest jurnal, putem recrea orice stare de joc la orice moment dat în timp. Acest lucru este deosebit de util pentru analiza ulterioară a jocului, deoarece ne permite să analizăm diferite scenarii și să identificăm unde un jucător a greșit sau a făcut o mișcare strălucită. De asemenea, putem folosi această istorie pentru a înțelege mai bine cum a evoluat strategia jucătorilor de-a lungul jocului.

Pe lângă înregistrarea mutărilor, în jurnalul de joc putem include și alte informații relevante, cum ar fi momentul în care a fost achiziționată o anumită îmbunătățire sau numărul de turnuri cumpărate. Acest tip de date poate ajuta la crearea unor modele predictive mai precise care pot informa despre deciziile viitoare ale jucătorilor. Mai mult de atât, prin analiza secvențelor de mutări și a deciziilor luate în cadrul jocului, putem folosi tehnici de învățare automată pentru a dezvolta strategii de joc mai eficiente.

În cazul nostru notația pentru un eveniment creat de jucător poate fi descris de următoarea structură:

- Numărul pasului la care a fost creat evenimentul.
- Tipul evenimentului (exemplu: cumpărare/mutare turn).
- Detalii despre eveniment (exemplu: tipul turnului, poziția).

Toate lista de evenimente va fi disponibilă în jurnalul de joc prezent în aplicație (Figura 24). Iată câteva exemple de evenimente însoțite de explicații care pot fi înregistrate în jurnalul de joc:

Exemplu 1: `#1 buy-tower a-crossbow@1 (2,0)` – la pasul 1, jucătorul a cumpărat un turn de apărare de tipul *crossbow* (turn activ prin prefixul *a-*) care a fost plasat pe poziția (2,0). Sufixul *@1* este identificatorul unic al turnului de apărare.

Exemplu 2: `#90 move-tower a-crossbow@1 (2,1)` – la pasul 90, jucătorul a mutat turnul de apărare de tipul *crossbow* de pe poziția (2,0) pe poziția (2,1). Poziția anterioră o deducem din evenimentele precedente.

Exemplu 3: `#150 buy-upgrade slow1` – la pasul 100, jucătorul a cumpărat îmbunătățirea pentru jetonul de încetinire (*slow*). Sufixul *@1* este identificatorul unic al îmbunătățirii în cadrul tipului de jeton.

Pe langă funcția de afișare avem și posibilitatea de a descărca jurnalul de joc în format *CSV (comma-separated values)* sau text simplu pentru a putea fi procesat în afara aplicației. Iar această evaluare a comenzilor poate fi realizată și cu tehnicii moderne de analiză a datelor precum *Python* sau *R*.

O funcționalitate care ar putea schimba complet modul cum abordăm echilibrarea mecanicii de joc este folosirea de tehnici de învățare automată pentru a crea modele predictive care să dezvolte strategii de joc. Aceste modele pot fi folosite pentru a crea agenți inteligenți care să joace jocul în locul jucătorului. Acești agenți pot fi folosiți pentru a testa echilibrarea mecanicii de joc de către designerul jocului. Mai mult de atât, acești agenți pot fi folosiți pentru a crea un sistem de asistență pentru jucători care să-i ajute să ia decizii mai bune în cadrul jocului – precum în șah prin intermediul unui motor de șah.

Având în vedere ultime avansuri tehnologice, putem merge și mai departe cu ideea prin crearea unui asistent virtual care să poată fi folosit pentru a înțelege mai bine mecanica de joc implementată. Folosind tehnologii precum *GTP-4* [33] putem crea un asistent virtual care să poată răspunde la întrebări despre joc. De exemplu, putem întreba asistentul virtual care este cea mai bună strategie de joc pentru un anumit nivel, sau cum să contracarăm un anumit tip de inamic. Puterea care ne oferă această tehnologie este impresionantă și cu siguranța va fi foarte explorată în viitorul apropiat.

Într-un joc posibilitățile sunt nelimitate și depind doar de creativitatea creatorului, iar prin aceste unelte putem explora mai multe posibilități și a alege cea mai bună soluție pentru jocul nostru.

Figura 24: Jurnal e joc.

5. EVALUARE

Aplicația prezintă toate functionalitățile necesare descrise în capitolele anterioare și poate fi folosită pentru a testa mecanica de joc.

Aceasta poate fi hostată pe un server web sau poate fi rulată local pe calculatorul personal. Prin hostare web o putem face disponibilă pentru oricine are acces la internet, iar prin rularea locală o putem folosi pentru a testa înainte de a-l publica. Vom măsura niște metrici pentru a vedea dacă aplicația este performantă ca aplicație web.

Vom folosii programul *Ligthouse* pentru o obține aceste metrici. Vom folosii măsurile pentru *Performance* (Performanță), *Accessibility* (Accesibilitate), *Best Practices* (Bune practici). Aplicația va fi hostată pe platforma Vercel prin intermediul funcționalității de publicare al

aplicaților realizate în *SolidJs* [34] care ne oferă un domeniu gratuit și un certificat *SSL* pentru a putea folosii protocolul *HTTPS*.

În urma testului am obținut următoarele rezultate:

- Performanță: 100
- Accesibilitate: 90
- Bune practici: 92

Alte metrici:

- *First Contentful Paint* (încărcarea primului paragraf de text sau imagine): 0,4 s
- *Speed Index* (viteza de încărcare a conținutului): 0 s
- *Largest Contentful Paint* (timpul de încărcare al celui mai mare element vizibil): 0,4 s
- *Time to Interactive* (timpul de încărcare al interfeței): 0,4 s
- *Total Blocking Time* (timpul de blocare al interfeței la încărcare): 0 ms
- *Cumulative Layout Shift* (mișcarea elementor vizibile în fereastra browserului): 0,001

Pe langă metrici, programul ne oferă și recomandări pentru îmbunătățiri, precum:

- Folosirea explicită a valorilor *width* și *height* pentru elementele de tip *img*.
- Utilizare atributului *alt* pentru a descrie imaginile pentru persoanele cu deficiențe de vedere.
- Imaginile folosite au o dimensiune prea mică (32x32 pixeli) față de cea recomandată (40x40 pixeli).
- Hostarea locală a fonturilor folosite pentru a reduce timpul de încărcare.

Folosind funcția de măsura a timpului *performance.now*, am calculat timpul de procesare al funcției principale de procesare al unui pas de joc. Am obținut următoarele rezultate din analiza a 382 de pași de joc:

- Timp minim: 2,90 ms
- Timp maxim: 14,11 ms
- Timp mediu: 8,27 ms

Timpul de procesare este foarte bun pentru nevoile aplicației, întrucât aceste nu va introduce un blocaj în timpul rulării pentru biblioteca *SolidJs* în actualizarea interfeței. *SolidJs* folosește o tehnică de actualizare incrementală a interfeței, iar acest timp de procesare nu va afecta performanța aplicației [35]. Acesta are o eficiență bună în comparație cu alte biblioteci de *UI* precum *React* sau *Vue* [36].

Cum nu avem procese grele de calcul precum calcule de rezolvarea coliziunilor sau de fizică, nu suntem nevoiți să folosim tehnici de optimizare a performanței precum *Web Workers* [37] sau *Web Assembly* [38].

Așadar, aplicația poate fi utilizată fără a întâmpina probleme de performanță.

CONCLUZII

În această lucrare am prezentat o idee de design de joc pentru genul *Tower Defense*. Am încercat să găsim o soluție pentru a rezolva problema repetitivității care apare în acest gen de jocuri. Am propus un sistem de jetoane de acțiune care să aducă o variație în joc prin faptul că jucătorul poate să combine diferite tipuri de turnuri de apărare pentru a crea jetoane de acțiune care să aibă efecte puternice asupra inamicilor. Iar acest sistem poate fi extins și asupra inamicilor prin faptul că aceștia pot reacționa diferit la anumite tipuri de jetoane de acțiune – fapt care oferă o nouă provocare jucătorului.

Creare unui design de joc este o sarcină dificilă întrucât trebuie ținem cont de numeroase aspecte care au rol important în experiența finală de joc. De exemplu, trebuie să ținem cont de: mecanică de joc, interfața de utilizator, tematica jocului, publicul țintă, platforma de lansare, monetizarea, etc.

Crearea de jocuri este un proces creativ care necesită multă experiență și multă muncă. De aceea, este important să avem o metodă de lucru care să ne ajute să ne organizăm și să ne concentrăm pe aspectele importante ale jocului. De aceea este bine să avem cât mai multe referințe pentru a ne inspira și pentru a ne ajuta să ne dezvoltăm abilitățile de design de joc.

Rolul unui prototip este de a ne ajuta să testăm ideile de design de joc într-un mod rapid și ieftin. Acesta ne ajută să ne dăm seama dacă ideile noastre sunt viabile sau nu. De asemenea, ne ajută să ne dăm seama dacă ideile noastre sunt interesante sau nu. Un prototip nu trebuie să fie perfect, ci trebuie să fie suficient de bun pentru a ne ajuta să testăm ideile de design de joc.

Aplicația dezvoltată în cadrul acestei lucrări prezintă un medium bun de prototipare a noilor jocuri bazate pe sistemul de jetoane de acțiune și oferă la rândul său unelte de analiză care facitilitează procesul de testare șă înțelegere a acestor noi idei.

Așa cum a fost prezentat în lucrare, design-ul flexibil oferă o varietate de opțiuni pentru a crea un joc. Dar un dezavantaj este că uneori nu avem nevoie de atât de multe opțiuni întrucât aceste pot complica în mod intenționat experiența de joc. De aceea, este important să avem un scop bine definit pentru jocul pe care îl creăm și să ne concentrăm pe acele aspecte care ne ajută să atingem acel scop.

Cu cât cunoaștem mai bine publicul nostru, cu atât putem crea un joc care să se potrivească preferințelor și așteptărilor acestuia. În cazul genului Tower Defense, publicul poate varia foarte mult în funcție de complexitatea jocului, de tematica acestuia, de platforma pe care este lansat, și așa mai departe.

Atunci când vine vorba de monetizare, trebuie să avem în vedere cât de dispus este publicul nostru să plătească pentru a juca jocul nostru. Monetizarea poate fi realizată prin mai multe modalități, cum ar fi achizițiile în aplicație, abonamentele, publicitatea sau vânzarea jocului la un preț fix. Monetizarea trebuie gândită în așa fel încât să nu împiedice experiența de joc și

să fie în acord cu așteptările publicului nostru. Natura extensibilă a sistemului de jetoane de acțiune permite cu usurință implementarea unui sistem de monetizare întrucât putem avea module de joc care pot fi achiziționate separat de către jucători.

În încheiere, putem spune că designul unui joc este un proces complex care necesită multă creativitate și pragmatism. Ideea de design de joc prezentată în această lucrare este doar un punct de plecare. Este important să fim deschiși la noi opinii și să fim pregătiți să adaptăm și să îmbunătățim ideile noastre pe măsură ce testăm prototipul și primim răspunsuri constructive de la publicul nostru.

BIBLIOGRAFIE

[1] L. Mitchell, "Tower defense: bringing the genre back." https://web.archive.org/web/20140203062250/http://palgn.com.au/11898/tower-defense-bringing-the-genre-back/ (Last accessed on 2023-06-09)

[2] 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.

[3] R. Lewis, "Tower defence games to reach $2.3 billion in revenue by 2030. " https://www.pocketgamer.biz/news/79629/tower-defence-games-to-reach-23-billion-in-revenue-by-2030/ (Last accessed on 2023-06-20)

[4] "Chess." https://www.chessprogramming.org/Chess (Last accessed on 2023-06-09)

[5] C. Browne, and F. Maire, "Evolutionary game design," *IEEE Trans. Comput. Intell. AI Games*, pp. 1–16, doi: 10.1109/TCIAIG.2010.2045292.

[6] 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.

[7] "Game design principles - tower defense." https://www.reddit.com/r/gamedesign/comments/j70hme/game_design_principles_tower_defense/ (Last accessed on 2023-05-27)

[8] D. Lars, "Optimizing tower defense for focus and thinking - defender's quest." https://www.gamedeveloper.com/design/optimizing-tower-defense-for-focus-and-thinking---defender-s-quest (Last accessed on 2023-06-10)

[9] S. Rogers, *Level Up! The Guide to Great Video Game Design*, John Wiley & Sons.

[10] P. Tassi, "Steam reveals its highest-grossing, most played games of 2022." https://www.forbes.com/sites/paultassi/2022/12/30/steam-reveals-its-highest-grossing-most-played-games-of-2022/ (Last accessed on 2023-06-20)

[11] N. Long, "2022's top grossing mobile games: Honor of kings, PUBG mobile, genshin impact and more." https://mobilegamer.biz/2022s-top-grossing-mobile-games-honor-of-kings-pubg-mobile-genshin-impact-and-more/ (Last accessed on 2023-06-20)

[12] "Merge games: to do or not to do? ." https://www.globalgamesforum.com/merge-games-to-do-or-not-to-do/ (Last accessed on 2023-06-10)

[13] E. Adams, and J. Dormans, *Game Mechanics: Advanced Game Design*, New Riders, 2012.

[14] "Kingom rush wiki -enemies." https://kingdomrushtd.fandom.com/wiki/Category:Enemies (Last accessed on 2023-06-10)

[15] "Kingdom wiki - coin." https://kingdomthegame.fandom.com/wiki/Coin (Last accessed on 2023-06-10)

[16] "Kingdom wiki - online co-op." https://kingdomthegame.fandom.com/wiki/Co-op#Online_co-op (Last accessed on 2023-06-10)

[17] J. Walcher, "Event-driven game engine in realtime strategy games."

[18] "Three ways to architect your game with scriptableobjects." https://unity.com/how-to/architect-game-code-scriptable-objects (Last accessed on 2023-05-27)

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

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

[21] T. Härkönen, "Advantages and implementation of entity-component-systems," 2019.

[22] "Unity's data-oriented technology stack (dots)." https://unity.com/dots (Last accessed on 2023-05-28)

[23] "Solidjs." https://www.solidjs.com/ (Last accessed on 2023-06-10)

[24] "Kingdom rush wiki - levels." https://kingdomrushtd.fandom.com/wiki/Category:Levels (Last accessed on 2023-06-12)

[25] "Typescript." https://www.typescriptlang.org/ (Last accessed on 2023-06-13)

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT press, 2022.

[27] "Arknight wiki." https://arknights.fandom.com/wiki/Arknights_Wiki (Last accessed on 2023-05-19)

[28] "Best practices for designing an effective user interface." https://www.gamesindustry.biz/best-practices-for-designing-an-effective-video-game-ui (Last accessed on 2023-05-20)

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

[30] Y. Xie, H. Hofmann, and X. Cheng, "Reactive programming for interactive graphics," *Statistical Sci.*, pp. 201–213, 2014.

[31] "React." https://react.dev/ (Last accessed on 2023-06-12)

[32] "Chess notation." https://www.chess.com/terms/chess-notation (Last accessed on 2023-06-12)

[33] OpenAI, "Gpt-4 technical report," *Arxiv*, 2023.

[34] "How to deploy a solid app with vercel." https://vercel.com/guides/deploying-solid-with-vercel (Last accessed on 2023-06-20)

[35] R. Carniato, “B.Y.O.F. — part 4: Rendering the Dom.” https://ryansolid.medium.com/b-y-o-f-part-4-rendering-the-dom-753657689647 (Last accessed on 2023-06-20)

[36] S. Krause, “Results for js web frameworks benchmark.” https://krausest.github.io/js-framework-benchmark/2023/table_chrome_114.0.5735.90.html (Last accessed on 2023-06-20)

[37] “Using web workers.” https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers (Last accessed on 2023-06-20)

[38] “Web assembly.” https://developer.mozilla.org/en-US/docs/WebAssembly (Last accessed on 2023-06-20)