Prototype analysis for Turn-based combat game

2281696, Robert Johannsen-Hanes 26/03/21

Intention

This is a analysis for a micro project focusing specifically on the data design of the prototype.

Concept

The prototype explores a turn-based combat system where stats effect everything. The combat system in the game is inspired by the 1st 2 Fallout games combat system where the player can queue attacks and target individual parts of an enemy – this prototype simplifies this system.

Data Design Process

Combat System

The combat system is defined by the stats the combatant has; the system was designed to be flexible with hard limits for future balancing purposes.

Each combatant has these base stats

HP: combatants health points

AP: the amount of action points and the cost of actions

Arm reduces the damage taken by the value of the armour

Att : combatants attack damage modifier

Crit: combatants base critical hit modifier

Agi: combatants agility modifier, determines who moves first in combat

Acc: combatants base accuracy modifier

Turn order

Beginning combat the game checks for the highest Agi combatant and gives the initiative. Then the game does a bunch of dice rolls first checking for if the hit lands using the base combatant accuracy (_accMod) as well as the base accuracy of the part they want to hit (_baseAcc) which adds up to a percentage shown to the player.

_finalAcc = Mathf.Clamp((_baseAcc + (_accMod * 2)), 0, 100)

If the hit lands it checks for a crit using the combatants crit chance (_critmod) and the attacks crit chance (_baseCrit). If the base attack has no chance to crit it cannot no matter the critmod. Crits multiply the final damage of the attack by 1.5.

_critChance = Mathf.Clamp(((_baseCrit / 100) + 1) * (_critmod + 25), 0, 100)

Damage

Damage in the game is calculated using the base attack damage (_baseAtt), the combatants attack modifier(_attMod) and critical hit modifier (_crit). The damage system was designed so that in theory the maximum amount of damage an attack can do is double its base attack. Lastly the targets armour value is subtracted from the damage.

_bonusdmg = (_attMod / 10) * (_baseAtt/2)

_dmg = (int)((_baseAtt + _bonusdmg) * (_crit))

_dmg = Mathf.Clamp((_dmg - _targetArm), 0, 200)

Other systems

With the ability to select what part to attack, each part has its own HP repeatedly damaging the part will cripple it and apply its unique effect (shooting the leg will cripple the combatant lowering their Agi). Each part has benefits and trade-offs to hitting them (shooting the hat of a combatant which has a low base accuracy to hit will stun them reducing their AP to 0).

AP

Each combatant has an action point pool which dictates the moves they can queue up per turn, each move has an ap cost, combatants have a base AP regeneration rate after each turn.

Reflection

The project landed up over scoping due to time constraints, where most of time was spent bug fixing the data systems (at this point it's a gamble if they'll work as intended).

Using the stat based system allowed for fast direct changes to balancing

The main problem for the prototype was time, time spent bug fixing could have been used to make a better system for presenting information

Most of the systems in the project were developed separately from each other and quickly stuck together with mixed results.

Better communication could've been used with the aiming system (like knowing what the health of each part was and what effect it had on destruction)

The damage system provides a neat foundation to build off with the flexibility of the stat system and attack system should further be developed. Using action point pool to chain attacks in each turn allowed for multiple decisions per turn but the development of it became a nightmare and if the project is returned to this needs a rework

Overall, the project has a good data system foundation but is brought down by how each system interacts with each other