DISPLAY

Design Specification

Version 1.5

Prepared by:

James Treneman

Prepared for:

Steve Sheehy

CS 401 Capstone

1. Table of Contents

2. Introduction

This design document details DISPLAY's implementation. Each section will help fill in the unknowns of how DISPLAY functions until there is a full picture, enabling productions tasks to be performed definitively.

3. User Experience (UX)

Definition of user interface (UI), intent behind the design, and desired user response. Each screen will be itemized and explained to establish the letter, and "spirit of the law".

3.1   UI Overview: DISPLAY is comprised of 3 game screens: Title, Select, and Battle. Game flow begins at Title, goes to Select, then to Battle. After a battle is concluded, the game goes back to Select (completing the loop).
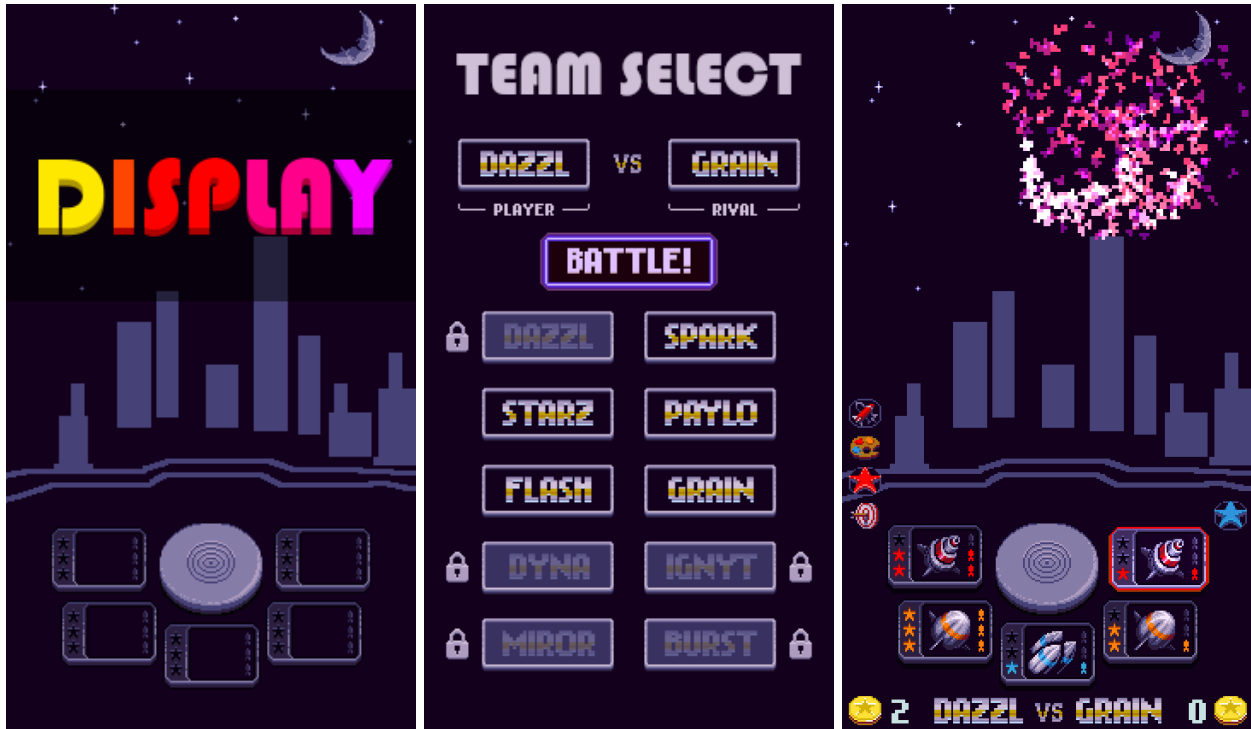
Figure 1) Title, Select, and Battle screens.

3.2   Title Screen: This screen simply shows the title and demonstrates a firework animation in the background to draw attention. DISPLAY inputs are limited to mouse left press / mouse left release, and spacebar press / spacebar release. Touch screen inputs are automatically translated into mouse left actions by the Integrated Development Environment (IDE). Any of these inputs will advance DISPLAY to the Select Screen and trigger the global cooldown (GCD) which prevents user input for a couple of seconds.
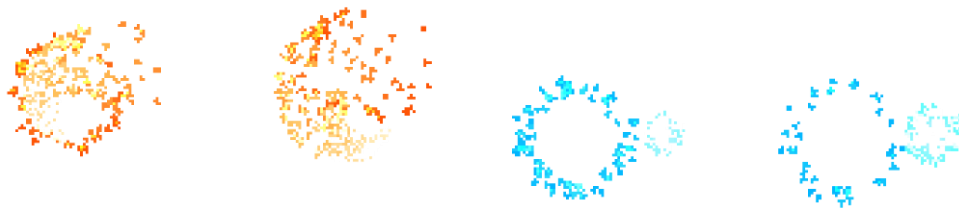
3.3   Select Screen – Versus Panel: Here is where DISPLAY's replay premise is conveyed. DISPLAY pits the player against a rival, each selected from 1 of 10 choices. Initially players can only choose to play AS the first team, and AGAINST the second team. However, once they win against a team, that team's lock is removed and becomes playable. Once the player selects their own team, they must pick a rival team which is usually the next undefeated team on the list. Besting all 10 teams in battle 'completes' the game.

Figure 2) Player team select, and teams locked in ready for battle.

3.4   Select Screen – Team Panel: A grid of buttons, each with a team name as a label. Teams that are locked have an overlay sprite with a lock to the side. To prevent mirror matchups, after the player selects their own team, that team gets locked out from rival selection. Right side lock sprites are flipped on x-axis.

Figure 3) Team panel with lock sprites, rival selection (player team locked out)

3.5   Battle Screen: Once battle is initiated, the game leaves the Select Screen and reveals the battle screen. The core user interface consists of the control panel on the bottom (large launch button and 5 inventory slots) and the bottom status bar with team names and star coin tallies.


Figure 4) Battle Screen, A) launch button, B) inventory slot, C) selected inventory slot, D) rival star coins, E) versus banner.

3.6   Battle Screen – Launch Button (A): This button turns red and pops up a few pixels when available for use. It is only usable beginning a few seconds after the rival's shot is fired. The player must have an inventory slot selected as well or the button will remain inactive.

3.7   Battle Screen – Inventory Slot (B): These 5 buttons represent the player's inventory of fireworks. Each has a power level, signified by lighting up 1, 2, or 3 stars, and the quantity remaining on the right conveyed via small ammunition icons. The shot's color is signified with color cues on the shot sprite itself along with the stars and ammo icons. The central icon in each slot represents the type of shot. The 3 types of fireworks are: mega (largest explosion), spinner (corkscrew animation), and multi (numerous smaller explosions). When a slot's ammo has been depleted, it's rendered unclickable and all that remains is an empty slot frame with no other icons.

3.8   Battle Screen – Selected Slot (C): Clicking on a slot activates it, turning on a highlight outline that matches the color of the slot's fireworks. For more visual impact, the sprite's glow function is activated, making it much brighter overall. A keypress sound also plays and the button animates downward a couple pixels.

3.9   Battle Screen – Star Coins (D): Whichever team wins an exchange (one shot from each team) earns a star coin. The left count is the player's and the right side covers the rival's total star coins. If the player has more star coins than their rival at the end of a match, the player is victorious. An even number of coins results in a tie; if the rival has more, the player loses. The winning team's coin descends from where the point icons are displayed down to their coin tally.

3.10   Battle Screen – Versus Banner (E): Displays what teams are selected for the player and rival.
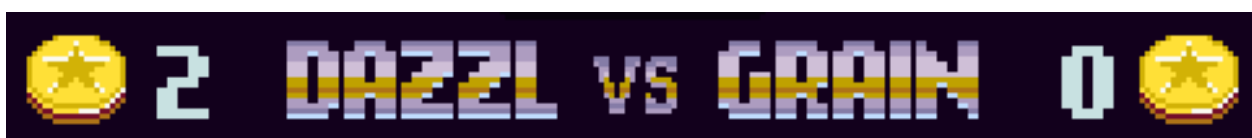
Figure 5) Versus banner and star coin tallies (left side player, right side rival).

3.11   Battle Screen – Launch Meter: Once the player presses the launch button, the launch meter appears. It starts as a single, colored bar just above the launch button, growing upward by stacking up 20 separate bars. At the very top, there is a special, large bar with a star cutout. The player needs to release their launch button exactly when the launch meter shows the top star bar to earn the accuracy bonus.

3.12   Battle Screen – Point Stack: Here, each shot is scored according to the judging rules. Corresponding icons for accuracy, power level, color bonus, and superior shot type are shown depending on how the two launches compare. These icons stack upwards so that their visual height naturally indicates who won. They begin to appear the instant that the player releases their shot, with launch precision and shot power being judged first. Once the shot explodes, color (paint palette) and shot type (crossed rockets) are compared, and the point winners see the relevant icons.
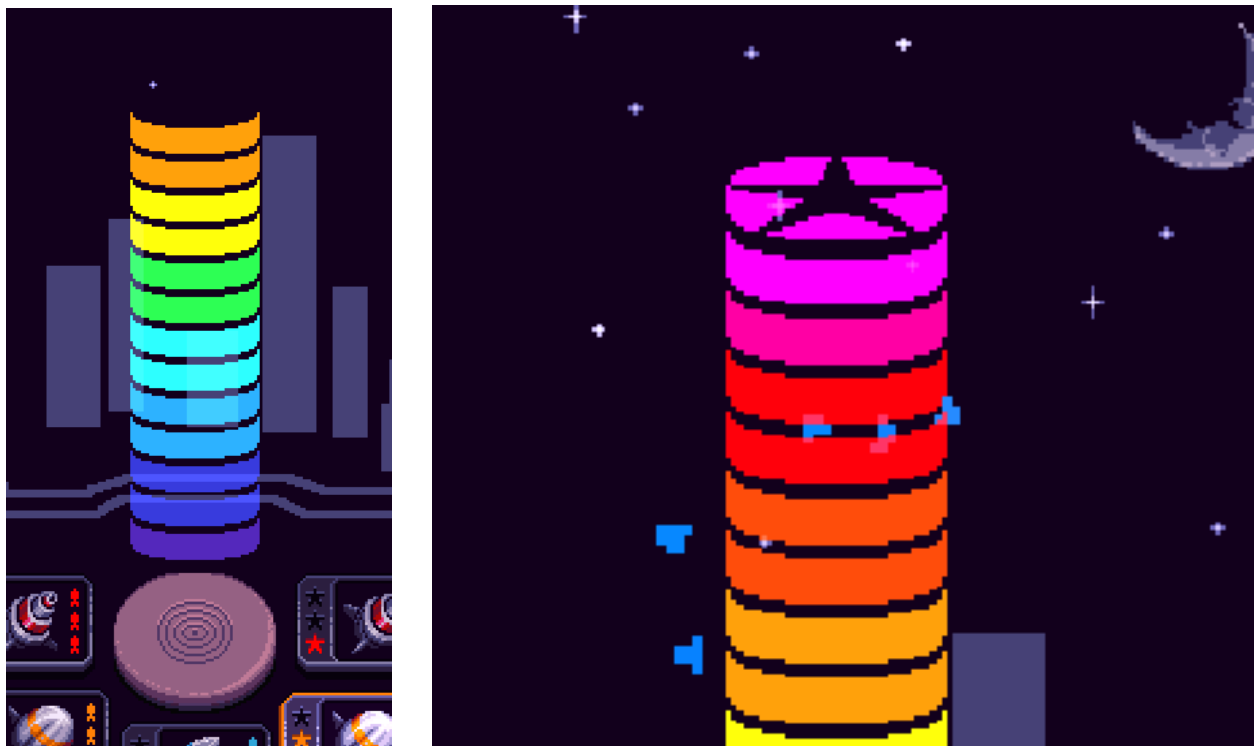
Figure 6) Rising Launch Meter and top star target.

Figure 7) Point stacks (left side player, right side rival) – this exchange was a draw.

4. Research

To better formulate DISPLAY's design, research has been done in the following arenas:

- Casual gaming demographics and market share:

  https://adjoe.io/glossary/casual-games-and-casual-gamers/

- HTML5 game production and deployment:

  https://elements.envato.com/learn/building-your-first-html5-game

- Gamemaker Studio 2 pricing models and IDE investigations:

  https://manual.gamemaker.io/monthly/en/Content.htm

- Review of popular 2023 casual puzzle games:

  https://www.rockpapershotgun.com/the-best-puzzle-games
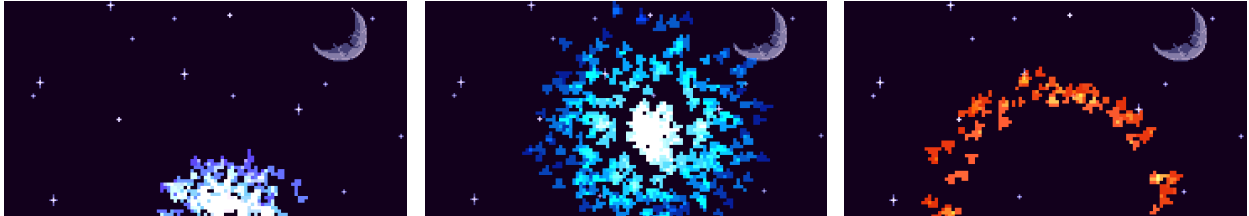
5.   Development Plan

5.1   **Concept / Placeholder Art:** UI mockups, determine aspect ratio, produce a few fireworks animations.

5.2   **Core Structure Implemented:** Battle screen playable and testable. Combat flow formalized, points system working.

5.3   **Game Flow Finalized:** Title, Select, and Battle screens present and navigable. Battle sequence explored and implemented.

5.4   **Playable Vertical Slice**: All phases of the game functional, tested, ready for iteration with additional assets.

5.5   **Visual Asset Production:** Complete a unified set of all art assets. Ensure a uniform level of fidelity and polish across the board.
   1.   Title screen image
   2.   Select screen button frames + lock image
   3.   Battle button + highlight
   4.   Team name sprites (10)
   5.   Battle screen UI components A, B, C, D, E
   6.   Point icons (6)
   7.   Coin icon
   8.   Freworks animations (3 colors x 3 types x 3 power levels = 27)
   9.   Win / Lose images

5.6   **Sound Asset Production:** Finish all necessary sound effects (sfx) and jingles.
   1.   Start game / win match jingle
   2.   Lose match jingle
   3.   Press button affirmative sfx
   4.   Press button negative sfx
   5.   Click mechanical button sfx

6. Press launch button sfx

7. Hold launch button loop

8. Perfect launch cue sfx

9. Cannon for launch sfx

10. Fireworks exploding sfx

11. Points popping onto screen sfx

12. Coin earned cue sfx

13. Exchange won jingle

14. Exchange lost jingle

15. Exchange tied sfx

5.7 **Playtest / Bugfix:** Balance gameplay, formalize difficulty curve. Get third-party testing and utilize feedback. Remove frustrations or conveyance issues. Analyze experience across intended platforms.

5.8 **Game Release:** Design an itch.io page for DISPLAY and publish. Generate necessary header, gameplay, and icon images. Write copy introducing game.

6. Design

6.1   Entity Relationship Diagram (page 11)

6.1   UML Diagram (page 12)

Ctrl_game

+ hud: Ctrl_hud
+ mode: 0=title, 1=select, 2=battle
+ locked: true=no interaction
+ unlock_num: teams unlocked (1-10)
- teams[]: selected teams for [0]=player,
[1]=rival (1- 10)
- player_inventory[][]: 1-5 shots available
for player to launch [x][], shot has type,
color, quantity, and power stats [][y]
- rival_inventory[][]: 1-5 shots available for
rival to launch, same as player's
- points[0]: [0]=player points, [1]=rival
points, [2]=player coins, [3]=rival coins
- shot_count[]: [0]=player shots, [1]=rival
shots (1-15, 5 fireworks, 3 quantity each)
- gcd: global cooldown (180, prevents
interaction for 3 seconds)

+ init(): setup all arrays and triggers
+ lock(): lockout interaction
+ reset(): reset all arrays and triggers
+ next(): advance between screens
+ judge(): compares both team's fireworks
+ rival_launch(): fires enemy shot

Ctrl_hud

- launch_button: Launch_button
- launch_meter: Launch_meter
- slot[]: Slot_button (5)
- team_turn: 0=player, 1=rival
- locations[][]: [x][y] sprites positions
- shown_points[]: 0=player points, 1=rival
points, 2=player coins, 3=rival coins
- teams[]: [0]=player, [1]=rival (subimage)

+ init(): setup all arrays and triggers
+ reset(): reset all arrays and triggers
+ draw(): all visuals drawn

Launch_button

- pressed: bool
- flip: int (pulsing effect)

+ status(): check if active
+ reset(): reset triggers

Obj_sprite

- glow: bool
- one_time: bool
- timed: bool
- flash: bool
- flash_time = 7

+ draw(): draws itself
+ flash(): use blend to blink
+ destroy(): based on bools

Launch_meter

- speed: float (meter growth)
- chip: float (acceleration)
- release_frame: int
- cd: pause between frames

+ fire(): visible, active, looping
+ reset(): reset triggers

Slot_button

- selected: bool
- updated: bool
- slot: int
- type: 1=mega, 2=spin, 3=multi
- dmg: shot power level (1-3)
- color: 1=blue, 2=orange, 3=red
- count: ammo remaining (3-1)
- shot_sprite: sprite to overlay

+ select(): checks if selectable, activates
+ pressed(): upon valid input and state

Flare

- spd: initial speed (10)
- one_time: bool
- side: flips all coords for rival
- lift: thrust (float)
- precision = random factor

+ destroy(): deletes self and makes
a firworks sprite