Game Design Test — Federico Di Benedetto

Part 1: Theory Test

Contents

1.1 What is game design?

My choices:

- Process of planning and implementing the rules and content of the game
- Process of generating ideas for new games

Explanation:

Game design combines ideation and practical system creation. It defines how the game works and feels. Tasks like graphics, sound or writing have their own roles, but they all follow the direction set by design.

1.2 Common responsibilities of a game designer include:

My choices:

- Producing and updating documentation
- Prototyping and testing new ideas
- Adjusting balance and difficulty curves
- Staying up to date on industry trends

Explanation:

Designers turn ideas into playable systems, document them clearly, iterate through prototypes and maintain balanced gameplay. Understanding industry practices keeps designs relevant.

1.3 Most important features of casual mobile games:

My choices:

- Ease of learning and comprehension
- Short and simple core loop
- Frequent rewards

Explanation:

Casual games rely on fast onboarding, short sessions and regular rewards to keep players engaged. Graphics and depth can be added later, but clarity and accessibility come first.

1.4 Which of these belong to core gameplay mechanics:

My choices:

- The skill system for the player character
- Conditions for winning or losing a game

Explanation:

Core mechanics define what the player does and what success means. Sharing features or settings options belong to meta systems, not core gameplay.

1.5 Most popular genres among female audience aged above 30:

My choices:

- Time management games
- Puzzle games
- Hidden Object Puzzle Adventure games

Explanation:

Research shows this demographic prefers accessible gameplay with progression, collection and light narrative elements instead of fast or reaction-based genres.

1.6 Improving poor long-term retention:

My choices:

- Explore loyalty rewards
- Add more depth to gameplay
- Review and adjust progression systems

Explanation:

Long-term retention comes from meaningful progression, fair difficulty and systems that reward returning players. Advertising or tutorial tweaks help early retention, but not long-term engagement.

Part 2: Practical Calculations

2.1 Lootbox Optimization

To understand which loot box is the cheapest way to get both the Sword and the Armor, I compare their probabilities using a simple expected-value approach.

Option 3: Box with 6 percent chance of giving both items
Cost: €0.70
Chance: 6 percent to obtain both at once
If I imagine buying 100 of these boxes, about 6 of them should contain the Sword and Armor together.
To get one successful drop, the expected number of purchases is:
100 divided by 6 ≈ 16.6 boxes
Cost: 16.6 × €0.70 ≈ €11.62
This is the expected cost of getting both items using only Box 3.

Option 1 + Option 2: Getting sword and armor separately
Sword (17 percent chance)
100 divided by 17 ≈ 5.88 boxes
Cost: 5.88 × €0.99 ≈ €5.82
Armor (25 percent chance using Box 2)
100 divided by 25 = 4 boxes
Cost: 4 × €1.20 = €4.80
Combined expected cost: €5.82 + €4.80 = €10.62

Final conclusion:

Buying Box 3 repeatedly costs about €11.6 on average to get both items. Buying Box 1 for the Sword and Box 2 for the Armor costs about €10.6 on average. Therefore, the most cost-efficient strategy is to use Box 1 for the Sword and Box 2 for the Armor.

2.2 Necromancer Damage Calculation

Clarification of interpretation:

The test wording states that the Necromancer can spend 10 percent of its current attack power to summon a Skeleton and that each subsequent Skeleton takes 10 percent of the Necromancer's remaining power. I interpret this as a one-time cost paid at each

summoning, not a continuous drain while skeletons are alive. All calculations below follow that interpretation, i.e. each summoning permanently reduces the Necromancer's attack power by 10 percent of its current value at the moment of summoning.

Step-by-step calculation:

Necromancer base attack power: 250
Attack speed: 0.5 hits per second
Base DPS: 250 × 0.5 = 125 DPS

Summoning sequence:

Skeleton 1 power = 10% of 250 = 25. Necromancer remaining = 250 − 25 = 225
Skeleton 2 power = 10% of 225 = 22.5. Necromancer remaining = 225 − 22.5 = 202.5
Skeleton 3 power = 10% of 202.5 = 20.25. Necromancer remaining = 202.5 − 20.25 = 182.25
Skeleton 4 power = 10% of 182.25 = 18.225. Necromancer remaining = 182.25 − 18.225 = 164.025

Final DPS values:

Necromancer DPS after reductions: 164.025 × 0.5 = 82.0125 DPS
Skeleton DPS: (25 + 22.5 + 20.25 + 18.225) × 0.3 = 25.8 DPS (rounded)
Total combined DPS: 82.01 + 25.8 ≈ 107.8 DPS

Time to defeat the Dragon:
Dragon HP = 5000
Time = 5000 / 107.8 ≈ 46 seconds

Final answer: It takes approximately 46 seconds for the Necromancer with four Skeletons to defeat the Dragon, using the interpretation described above.

Part 3: Game Analysis — Candy Crush Saga

Choose a free to play mobile game

I choose Candy Crush Saga because I am familiar with it and because it is one of the most successful mobile games for match three design and monetization.

Core loop description

Start game → Match candies → Trigger cascades → Finish level objectives → Earn stars and rewards → Unlock new levels on the world map → Receive lives and daily rewards → Repeat.

Short plain description

The player starts a level, swaps adjacent tiles to create matches of three or more, which clears tiles, triggers cascades and produces special tiles. Each level has a specific objective and a limited number of moves or time. Succeeding yields stars and rewards which feed the map progression and event systems. The loop is intentionally short and satisfying to allow multiple sessions per day.

Core mechanics used in the game

1. Tile swapping and match detection that triggers cascades and creates special tiles.
2. Move limited levels that require completing objectives with a fixed number of actions.
3. Special tiles and combination effects that change board state and create satisfying feedback loops.
4. Boosters and power ups that modify state and are purchasable or earned.
5. Progression via a world map with unlocked levels and new mechanics introduced progressively.
6. Lives system that gates sessions and creates natural cadence for return.

Monetization system and main principles of implementation

Candy Crush monetizes through microtransactions for extra moves, boosters, and lives refills. The core design principle is to keep new players engaged with accessible content and then create occasional friction points where a small purchase offers immediate relief. Limited time offers, event passes and cosmetics are used to generate spikes in revenue while live events and meta systems sustain long term spending. The economy balances frustration and reward so purchases feel optional but attractive at key difficulty moments.

Player progression system and how it is achieved

Progression is linear and level based with difficulty ramping over time. Levels award stars, occasional in level currencies or items, and unlock the next areas on the world map. New mechanics are introduced in controlled teaching windows inside levels so the player learns by playing. Event levels and timed challenges provide short term goals and alternate progression that complement the main map.

Three improvements I would introduce

1. Add a persistent meta progression that rewards repeat play with incremental power ups or a non-pay to win prestige system. This provides a long-term sense of growth that is independent from star collection and helps retention beyond the level list.

2. Introduce limited cooperative events where teams of players contribute to shared objectives. This adds light social pressure and motivation without requiring synchronous play and is likely to improve retention for mid core players.

3. Incorporate a simple puzzle editor and sharing mechanism so players can create, share and rate levels. Community created content increases retention and creates organic discovery.

Follow up notes to other teams and order by effort

1. Live ops request. Design event templates, economy parameters for cooperative events and pass rewards. Effort estimate high.

2. Engineering request. Extend server support for team events, event scoring and sharing user generated levels. Effort estimate high.

3. UX request. Design editor UI for puzzle creation and sharing flows including moderation entry points. Effort estimate medium.

4. QA request. Test user submitted content moderation, edge cases in sharing and event scoring. Effort estimate medium.

5. Narrative request. Create event flavor text and personas for team events to give context and voice. Effort estimate low.

Part 4: Game Balancing — Match-3 Puzzle Approach

Chosen game

I choose **Candy Crush Saga**, one of the most recognizable Match-3 puzzle games on mobile. Since I already analyzed it in Part 3, using the same title again helps me stay coherent in my reasoning. It was also easier to complete both sections with a single game because it fits perfectly for both.

What I would start with when balancing the game

I begin by checking basic player performance data to understand where the difficulty behaves as expected and where it does not. In Match-3 design, randomness plays a big role, so interpretation of the data is essential.

The first metrics I look at are:

- Completion rate for each level
- Number of retries players usually need
- How often boosters are used to finish the level

Then I compare this with the original design intention of the level. If a supposedly simple level ends up among the hardest ones, it becomes a clear balancing target. I also check **where new mechanics appear for the first time**, because players need a small learning moment before the difficulty increases again.

What the difficulty curve should look like and why

The difficulty curve in a casual Match-3 should feel fair and predictable. I describe it in simple stages rather than using heavy terminology:

- The first levels should be very easy, with the single purpose of teaching how the game works.
- Then the difficulty should rise slowly as the player unlocks new mechanics.
- Whenever a new mechanic is introduced, there should be a short “breather” where difficulty slightly drops so players can understand how the mechanic behaves.
- After the breather, challenge should increase again at a steady pace.
- The highest complexity should appear later in the game, where players are more experienced and ready for it.

This approach respects player learning and avoids sudden spikes that feel unfair.

How I would achieve that curve and what variables I would use

Match-3 games have a set of parameters that directly influence difficulty. The main variables I adjust are:

- **Move count**
- **Number and durability of blockers**
- **Board shape and available space**

- **Special tile generation frequency**
- **Number of colors present on the board**
- **Objective requirements such as jellies or ingredients**

Examples:
If a level feels too “stiff”, I can add a few moves or reduce blocker layers.
If the level feels random and inconsistent, lowering the number of colors helps generate more reliable matches.

These small adjustments help refine the experience without redesigning the whole level.

How I would evaluate the resulting balance

Once the adjustments are made, I evaluate the new version through both internal tests and A B tests.

I mainly look at:

- First-try success rate
- Average number of attempts before success
- Booster usage changes
- Short-term retention differences

If the new balance aligns better with the intended difficulty and improves fairness without harming monetization or retention, I apply the changes permanently. If not, I iterate again. For live games, balance is a continuous process that evolves with player behavior.

Part 5: UI Analysis — HOPA Settings Menu

The settings menu is considered final, so my evaluation focuses on usability, clarity, and how well it fits iPhone UI expectations. Since HOPA games typically attract a broad age range, including older players, readability and comfortable interaction are especially important.

1. How well is the UI optimized for iPhone?

Overall, the layout appears clean and understandable. Sliders and toggles are familiar components for iOS users, and the categories are easy to identify. The menu is also straightforward, which works well for a settings screen.

However, a few elements look slightly compressed for iPhone use:

- Some touch targets seem small for single-handed use on smaller models.

- Labels and interactive controls are placed very close vertically, which may be less comfortable on narrow screens.
- Some font sizes appear on the smaller side and could be difficult for older players.

These are not major issues, but they affect comfort and usability on iPhone devices.

2. Design defects or inconsistencies

Nothing drastically wrong, but I noticed a few small inconsistencies:

- Some icons seem slightly mismatched in style or weight.
- Certain settings appear visually close together, which can lead to accidental taps.
- Contrast between some elements and the background could be higher for better readability.

These are small polishing points rather than structural problems.

3. Improvements I would suggest

These suggestions are small, realistic improvements that respect the "final" state of the design:

- **Increase tap target size.** Every interactive element should match iOS recommended minimum touch areas, which helps players avoid mis-taps.
- **Increase spacing between rows.** This makes the layout more comfortable for thumb navigation and reduces accidental selections.
- **Use consistent iOS-friendly icons.** Ensuring all icons share the same visual style improves the perceived quality of the UI.
- **Improve contrast.** HOPA players often play in low light conditions, and higher contrast increases readability and comfort.
- **Add localization safety margins.** Some languages like German or Spanish take more space; the UI should not break when translated.
- **Add brief tooltips or short descriptions.** Some settings could benefit from a line of explanation to help less experienced players understand the options.

Part 6: Idea Presentation — Time Management Game for Male 30+ Audience

Chosen option

Chosen option: Time Management game for male 30+ audience.

Title: Garage Empire

Pitch: The player starts with a small, single-bay auto repair shop and grows it into a multi-location garage network. The gameplay mixes classic time management actions with a tycoon-style long-term progression system. Players accept repair jobs, assign mechanics, complete tasks under time pressure, earn money and reputation, and upgrade tools and locations. Daily events, prestige resets and seasonal content keep progression long-term. Core loop: Accept job → Diagnose → Assign mechanic → Complete repair → Earn money and rating → Upgrade → Unlock new districts → Repeat.

Core mechanics:

- Job queue with cars arriving based on time or events
- Mechanics with different specializations and skill levels
- Upgrades for tools, garage space, waiting area and car lifts
- Limited-time contracts with better rewards
- Reputation system that unlocks new districts and job types
- Daily tasks and rotating events

Meta progression / Long-term systems:

- Unlock new garages in different districts
- Prestige system with permanent efficiency boosts
- Seasonal events and collectible rare tools

Monetization anchors:

- Cosmetic skins for tools, mechanics and garage themes
- Speed-ups for long repair jobs
- Premium mechanics with unique buff specialties
- Event pass with exclusive rewards

Cross-team considerations:

Engineering: async job timers, upgrade database, district unlocking flow
Art: car silhouettes, repair animations, UI icons, district themes
Narrative: light flavor text for events and customers
QA: timers, job assignment logic, progression pacing
Live Ops: events calendar and seasonal tuning

Part 7: Design Documentation — Space Travel Game

Combat system: Real-time combat, suitable for dogfights with active aiming and module use.

Damage calculation: Damage = Weapon Power × Accuracy × Critical Modifier − Target Defense. Shields absorb damage before hull.

Win/loss: Win when enemy hulls reach zero. Lose when player hull reaches zero. Fleeing allowed but penalized.

Basic stats:

- Hull: permanent health, bar and number
- Shields: regenerating buffer, bar
- Energy: resource for modules, bar and regeneration indicator
- Speed: affects movement and evasion
- Firepower: combined offensive estimate
- Cargo capacity: numeric

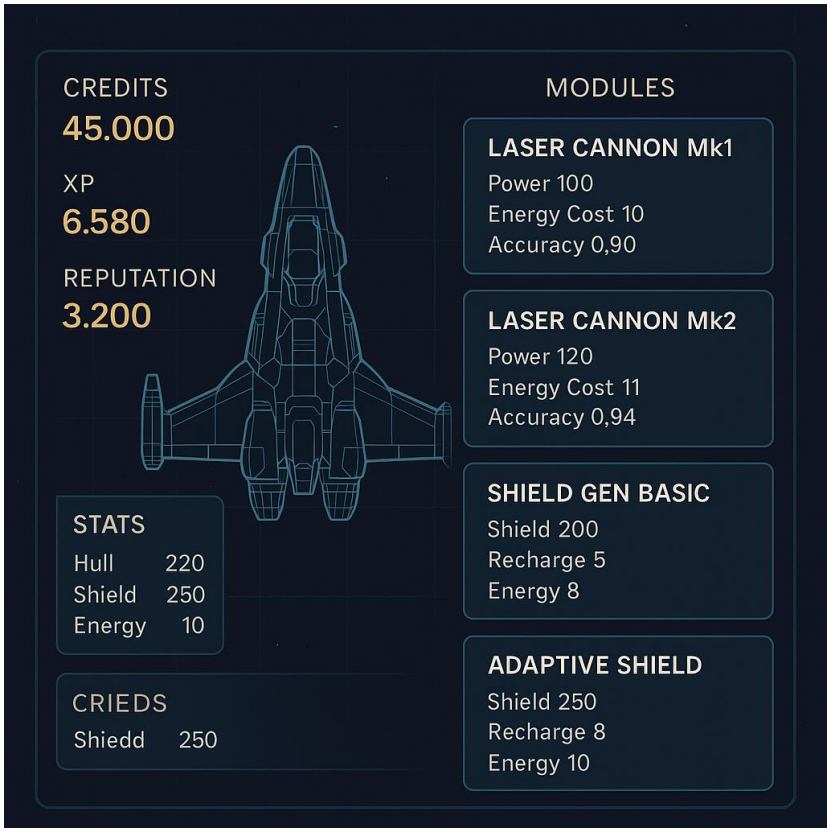
Module list and effects:

- Weapon modules: increase firepower or add effects
- Shield generators: increase shield or recharge
- Reactors: increase energy or regen
- Engines: increase speed and evade
- Armor plating: increases hull at cost of speed
- Cargo modules: increase storage
- Utility modules: scanners, repair drones, targeting assists

Progression system and examples:

Progression uses credits, XP and faction reputation to unlock module tiers. Example modules:

- Laser Cannon Mk1: power 100, energy cost 10, accuracy 0.9
- Laser Cannon Mk2: power 120, energy cost 11, accuracy 0.94
- Shield Gen Basic: shield 200, recharge 5, energy 8
- Adaptive Shield: shield 250, recharge 8, energy 10



Prototype priorities:

1. One playable ship with hull, shields and energy
2. One enemy ship with basic AI
3. One weapon, one shield and one engine module
4. Basic mission flow and rewards

Configuration example (short):

{ "modules": [{ "id": "laser_cannon_mk1", "type": "weapon", "power": 100, "energyCost": 10 }, { "id": "shield_gen_basic", "type": "shield", "shieldValue": 200, "rechargeRate": 5 }] }

Progression Tree

Laser Mk I	Shield Gen I	Engine Booster
Laser Mk II	Adaptive Shield	Engine Booster Mk II
Laser Mk III		Speed Drift Mk II

(Arrows implied: Laser Mk I → Mk II → Mk III; Shield Gen I → Adaptive Shield; Engine Booster → Engine Booster Mk II and Engine Booster → Speed Drift Mk II)

Part 8: Redesigning Chess

The goal is to redesign chess so that it includes dice and supports up to four players at the same time. My approach keeps the core rules recognizable while introducing simple, exciting mechanics that work well in a multiplayer context.

Board and player setup: 10×10 board, players at North, South, East and West. Each player starts with one King, one Rook, one Bishop, one Knight and four Pawns. Pawns move toward the center. Turn order rotates clockwise.

Dice system:

Roll two six-sided dice each turn. If the roll qualifies for a Power Action, the player may choose it.

Qualifying rolls:

total 7, doubles, or total 10 or more.

Power Actions:

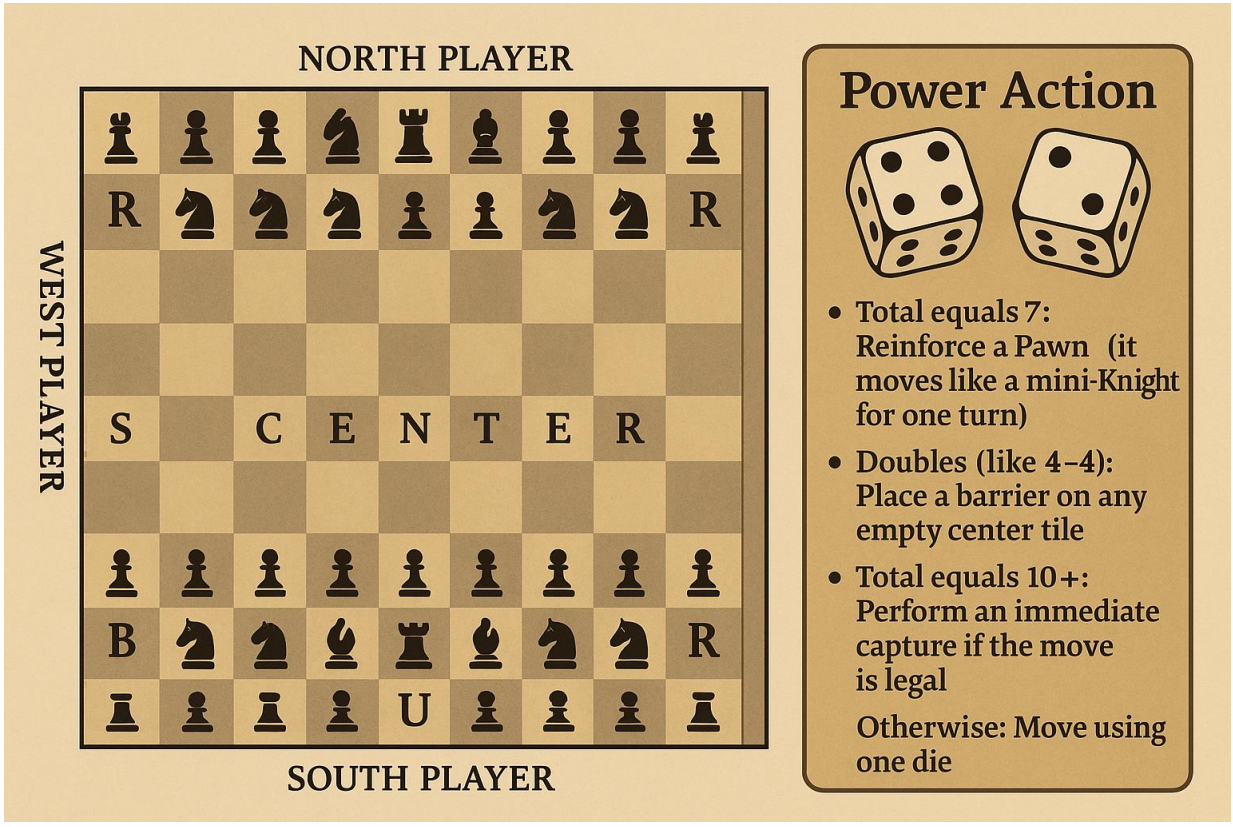
- total 7 reinforces a pawn for one turn;
- doubles place a temporary barrier in the center;
- total 10+ allows one immediate legal capture.

If the roll does not qualify, the player must choose one dice and use it to determine which piece can move: 1 or 2 Pawn, 3 Knight, 4 Bishop, 5 Rook, 6 King or any piece.

Movement and combat rules: Classic chess movement rules apply. No castling. Capture removes pieces as normal. Player eliminated when their King is captured; their pieces are removed.

Win conditions: Elimination victory is last King standing. Control victory is if one player controls the majority of central squares after 30 rounds.

Mock-up:


NORTH PLAYER
WEST PLAYER
SOUTH PLAYER
R
S
C E N T E R
B
U
R
Power Action
• Total equals 7: Reinforce a Pawn (it moves like a mini-Knight for one turn)
• Doubles (like 4–4): Place a barrier on any empty center tile
• Total equals 10+: Perform an immediate capture if the move is legal
Otherwise: Move using one die