Game Design Extractor

1. Objective & Importance

Objective:
To develop a prototype AI pipeline that analyzes curated visual game inputs (screenshots, short video clips) to extract and structure core game design elements. The system will output detailed, engine-agnostic JSON, specifying:

- **Visual Asset Descriptions:** Textual prompts for generating 3D/2D models (e.g., character appearance, object style).
- **Behavioral Specifications:** Logic for entities (player, enemies, obstacles) in a format translatable to behavior trees or state machines (e.g., "IF condition THEN action").
- **Spatial & Kinematic Data:** Estimated positions, scales, and simple movement patterns.
- **Game Mechanic Parameters:** Key values defining gameplay interactions (e.g., damage, health, spawn rates, UI-indicated values).

---

2. Scope & Assumptions

- **Input Focus:** The system will primarily process *curated and potentially pre-annotated* screenshots and short (5-15 second) video clips from a limited set of 2-3 distinct, visually clear game examples (e.g., Archero-like top-down shooter, simple 2D platformer).
- **Analysis Depth:** Prioritize defining a robust JSON schema and demonstrating the LLM's capability to populate it based on (potentially idealized) visual cues. Deep, perfect visual analysis across any game is out of scope.
- **Engine Agnostic:** All outputs and internal representations will be strictly engine-agnostic (primarily JSON and Python). No Unity or C# development - even though eventually it will be integrated into a Unity system

- **Behavioral Abstraction:** Behaviors will be described as logical rules or condition-action pairs, not as fully implemented behavior trees

---

3. Core Workflow

1. **User Input:**
    - Developer provides a curated game screenshot or a short video clip.
    - Optional: Simple annotations highlighting key entities or events (e.g., bounding boxes, event labels for initial bootstrapping).
2. **System Response (Analysis Pipeline):**
    - **A. Visual Parsing Module:**
        - Identifies key game entities (player, enemies, projectiles, collectibles, UI elements like health bars).
        - Extracts basic visual properties: estimated screen position, approximate size, dominant colors.
        - (For video): Tracks entities across frames to infer basic movement (e.g., "object A moved from X1,Y1 to X2,Y2")
    - **B. LLM Interpretation & Abstraction Module:**
        - Receives structured data from the Visual Parsing Module.
        - **Visual Description Generation:** Generates textual prompts describing the visual appearance of entities (e.g., "A green, orc-like creature with four arms and red eyes").
        - **Behavior Inference:** Translates observed actions and interactions into logical rules or state descriptions (e.g., "Enemy patrols between point A and B. IF player_is_close AND player_is_visible THEN state_becomes_CHASING_PLAYER").
        - **Parameter Extraction:** Identifies and quantifies game mechanic parameters (e.g., "Player_attack_animation observed; UI_health_bar_enemy decreased by 20 units → inferred_player_damage: 20").

- **Spawn/Event Logic:** Infers simple event-driven logic (e.g., "Coin appeared after enemy_X was destroyed → inferred_spawn_rule: coin_on_enemy_X_defeat").

- **C. Output Generation:**
  - Constructs a comprehensive JSON document based on the interpretations, adhering to a predefined schema.

---

4. Key Capabilities

- **Visual Detailing:** Generate descriptive text prompts suitable for guiding asset generation for 3-5 distinct entity types.
  - *Example: Input (simplified visual features of an Archero enemy) → Output: "A small, round, red demonic creature with short horns and glowing yellow eyes."*
- **Behavioral Description Generation:** For 3-5 distinct entity behaviors, generate clear, concise natural language descriptions of their core logic, triggers, and actions.
  - *Example: Input (enemy moves, player approaches, enemy turns and shoots) → Output: "This enemy remains stationary until a player comes into its line of sight and within a 10-meter range. Upon detection, it fires a single projectile at the player, then waits for 2 seconds before firing again if the player is still a valid target."*
- **Parameter Identification:** Extract and quantify 5-10 key game parameters (e.g., movement speed, attack damage, health points inferred from UI changes, collectible value).
  - *Example: Input (Whenever the orc enemy attacks, he does damage equal to 1/4th of the player's haelth)*
- **Structured Output:** Consistently populate a well-defined JSON schema with the extracted information, including the natural language behavior strings.
- **Placement Data:** Provide estimated 2D or simplified 3D coordinates (e.g., `[x, y, z_layer]`) and `scale` for key entities.

---

5. Output Format Spec (Illustrative Snippets - Behavioral Part Modified)


```
{
  "game_source_id": "archero_clip_01",
  "analysis_timestamp": "2024-03-15T10:00:00Z",
  "global_inferences": {
    "inferred_genre_tags": ["TopDownShooter", "ActionRPG"],
    "camera_perspective_approx": "TopDown_SlightlyAngled"
  },
  "scene_elements": [
    {
      "element_id": "player_character_01",
      "element_type": "PlayerCharacter",
      "visual_description_prompt": "A heroic archer figure, wearing a green tuni
c and a hood, carrying a bow. Agile stance.",
      "estimated_transform": {
        "position_2d_normalized": [0.5, 0.2],
        "scale_approx": [0.05, 0.1]
      },
      "behavior_description_prompt": "The player character is controlled by use
r input for movement. The player can initiate a ranged attack with their bow ap
proximately every 0.7 seconds. Player movement appears agile and responsiv
e.",
      "game_mechanic_params": {
        "inferred_player_attack_damage": 20,
        "max_health_inferred_from_ui": 100
      }
    },
    {
      "element_id": "enemy_melee_grunt_003",
      "element_type": "Enemy",
      "subtype_inferred": "MeleeGrunt",
      "visual_description_prompt": "A bulky, red-skinned demonic minion with t
wo large pincers for hands. Slow but determined movement.",
      "estimated_transform": {
```

```
      "position_2d_normalized": [0.7, 0.6],
      "scale_approx": [0.08, 0.08]
    },
    "behavior_description_prompt": "This enemy patrols slowly between two d
esignated points (e.g., point A at [0.6,0.6] and point B at [0.8,0.6]). If a player c
haracter enters its visual range (approx. 5 units) and is in its line of sight, it will
stop patrolling and charge directly towards the player. Once within close mele
e range (approx. 1 unit), it performs a melee attack. After attacking, it may brie
fly pause before re-evaluating or attacking again if the player is still in range.",
    "game_mechanic_params": {
      "health_observed_events": [{"event": "TOOK_DAMAGE", "amount": 20, "ti
mestamp_sec": 5.2}],
      "on_defeat_spawns_inferred_description": "Upon defeat, this enemy type
typically drops 2 to 4 gold coins.",
      "inferred_melee_attack_damage": 15,
      "inferred_movement_speed_patrol": 1.0, // units/sec
      "inferred_movement_speed_chase": 2.0 // units/sec
    }
  },
  {
    "element_id": "collectible_coin_010",
    "element_type": "Collectible",
    "subtype_inferred": "Currency",
    "visual_description_prompt": "A shiny, gold coin, flat, with a star insignia. R
otates slowly on its Y-axis and bobs gently up and down.",
    "estimated_transform": {
      "position_2d_normalized": [0.75, 0.65],
      "scale_approx": [0.02, 0.02]
    },
    "behavior_description_prompt": "This coin is a collectible item. It remains s
tationary, rotating and bobbing for visual effect. It disappears when the player
character moves close enough to touch it (pickup radius appears to be about
0.5 units).",
    "game_mechanic_params": {
      "value_inferred": 1
    }
```

```
    },
    {
      "element_id": "obstacle_turret_001",
      "element_type": "Obstacle",
      "subtype_inferred": "StaticTurret",
      "visual_description_prompt": "A small, metallic, stationary turret embedde
d in the ground. It has a single barrel that swivels.",
      "estimated_transform": {
        "position_2d_normalized": [0.3, 0.8],
        "scale_approx": [0.04, 0.04]
      },
      "behavior_description_prompt": "This turret is stationary. It periodically (e.
g., every 3 seconds) fires a single projectile in a fixed direction or towards the
player if the player is detected within its firing arc and range. It does not move
from its deployed position.",
      "game_mechanic_params": {
        "inferred_projectile_damage": 10,
        "firing_interval_seconds_approx": 3.0
      }
    }
  ],
  "ui_elements_observed": [
    {
      "ui_element_id": "player_health_bar_01",
      "inferred_function": "PlayerHealthIndicator",
      "screen_position_normalized_bbox": [0.1, 0.9, 0.3, 0.05],
      "observed_changes": [{"value_change_percentage": -20, "timestamp_s
ec": 5.2, "triggering_event_inferred": "ENEMY_ATTACK_HIT_PLAYER"}]
    }
  ]
}
```

Structure Explanation (Key Change Highlighted):

game_source_id, analysis_timestamp: Metadata.

global_inferences: Scene-wide observations.

scene_elements: Array of all significant identified entities.

element_id, element_type, subtype_inferred: Identification.

visual_description_prompt: Text for an image generation model or human artis t.

estimated_transform: Approximate spatial data.

behavior_description_prompt: A natural language string describing the entity's behavior. This string is intended to be a prompt for a future system that gener ates actual behavior logic (e.g., behavior trees, scripts).

game_mechanic_params: Quantifiable game design values.

ui_elements_observed: Information about UI components and their inferred pu rpose/changes.

Deliverables

1. **Source Code:** Python-based analysis pipeline, including all modules developed.

1. **JSON Schema:** Formal definition of the output JSON structure (e.g., as a JSON Schema file or clearly documented Python dataclasses).
2. **Example Data:** A set of 3-5 curated input examples (screenshots/short clips or paths to mock data) and their corresponding generated JSON output files.
3. **Technical Documentation:**
    - System architecture overview.

- Detailed explanation of the JSON schema fields.
- Key LLM prompting strategies and examples, especially for generating effective behavior_description_prompt strings.
- Instructions for setting up and running the system.

4. **Basic Test Suite:** Scripts to validate JSON outputs and test core LLM interpretation logic against predefined inputs/outputs.