**Idle Game Maker: Comprehensive Guide**

**WHAT IS IDLE GAME MAKER (IGM)** Idle Game Maker (IGM) is a web-based tool developed by Orteil and Opti (creators of Cookie Clicker) that allows users to create idle/clicker-style games using a structured plain-text file. It requires no traditional coding knowledge.

* **Purpose**: Create browser-based idle games.
* **Method**: Text file parsed by IGM’s engine.
* **Target Audience**: Game designers, hobbyists, and anyone interested in incremental game development.

**HOW TO MAKE A GAME**

1. **Write a Source File**
   * Plain .txt format.
   * Must begin with Let's make a game!.
   * Use IGM’s specific syntax to define your game.
2. **Host Your File Online**
   * Pastebin.com recommended.
   * Must be public/unlisted and set to never expire.
   * Only .txt files are accepted.
3. **Play Your Game**
   * Format: https://orteil.dashnet.org/igm/?g=FILE\_URL
   * Example: https://orteil.dashnet.org/igm/?g=9HYzgPWX for Pastebin ID.

**EXEMPLAR STRUCTURE: BUNNY CLICKER** The game “Bunny Clicker” serves as a complete demonstration of IGM features. This includes layout control, resource generation, click buttons, dynamic tooltips, upgrades, achievements, effects, and randomness.

By studying its structure, you can:

* Understand how to combine resources, buildings, and upgrades.
* Use on click effects with animation and conditional logic.
* Create tooltips, icons, and CSS-styled elements.
* Build progression through req: conditions.
* Structure a layout using default or custom boxes.

**CORE SECTIONS IN A SOURCE FILE**

### 1. Let’s Make a Game!

Defines game metadata.

* name: - Title of your game.
* author: / by: - Your name.
* desc: - Description text (can include HTML-like tags like <//>).
* created: / updated: - Dates.
* version: - Game version number.

### 2. Settings

Customizes background visuals and game behavior.

* background: / tiling background: - URL to background image.
* building cost increase: - % inflation on repeated purchases.
* building cost refund: - % refunded on selling.
* spritesheet: - Sprite sheet (name, dimensions, URL).
* stylesheet: - CSS file for styling (e.g., stuff/bigBlue.css).
* no particles, no bulk particles - Visual toggles.

### 3. CSS

Customize game UI/UX with standard CSS syntax. Used to:

* Style layout boxes, icons, tooltips.
* Create visual feedback (e.g., glow, wobble).
* Add animations via classes (e.g., .hasFlares, .shadowed).

### 4. Layout

Controls visual structure.

* use default applies a sensible auto-layout.
* Custom layouts define nested boxes with contains:.

### 5. Buttons

Clickable objects like Make a Bunny.

* on click: defines what happens.
* Animations like anim icon wobble add flair.
* Complex logic like if (have goldenTouch and chance(1%)) yield 1 goldenCarrot is supported.
* class: and icon class: control appearance.

### 6. Resources

Used to buy things or represent values.

* Can show earned, max, or be hidden when 0.
* Examples: \*bunny, \*goldenCarrot.

### 7. Buildings

Generate resources passively.

* on tick: yields resources every second.
* req: defines unlock conditions (e.g., 100 bunnies:earned).
* Example: \*meadow yields 90 bunnies/sec.

### 8. Upgrades

Apply passive bonuses or unlock features.

* passive: affects other things.
* May use tags, multiple icons, and requirements.
* Example: passive:multiply bunny yield of bunnyButton by 2.

### 9. Achievements

Triggered by milestones.

* req: checks click counts, resource thresholds, etc.
* Achievements can apply effects or just be collectible.
* Example: req:10 bunnyButton clicks.

### 10. Shinies

Timed elements like \*luckyBunny that appear randomly.

* Have movement effects (e.g., moveLeft, fade, bounce).
* Can give rewards based on random outcomes.
* May reference upgrades to enhance effect (e.g., clover).

### 11. Templates & Includes

* Use \*TEMPLATE to apply shared settings (e.g., animations).
* \*include blocks can be reused to reduce repetition.

**COMMON FEATURES IN “THINGS”**

* name: - Single or dual form (e.g., Bunny|Bunnies).
* desc: - Description with optional formatting (<b>, <.>, <//>).
* icon: - From spritesheet or individual file.
* class: - CSS classes like bigButton, shadowed, noBackground.
* req: - Conditions to unlock or activate.
* cost: - What it takes to buy.
* on tick: / on click: / passive: - Game logic.

**EFFECTS (LOGIC ACTIONS)**

* on click: – Clicking behavior.
* on tick: – Passive per-second effects.
* passive: – Always-active logic (for upgrades).
* do effect – Trigger custom logic.

**Logic Examples:**

on click:yield 1 bunny  
on tick:yield 90 bunnies  
on click:if (chance(10%)) yield 1 goldenCarrot

**EXPRESSIONS & CONDITIONS**

* Arithmetic: +, -, \*, /, %
* Random: random(X), chance(X%), frandom(X,Y)
* Logic: have X, no X, and, or, if (X)
* Resource access: bunnies:ps, bunnyButton:clicks

**Sample Requirement:**

req:(pens>=50 or meadows>=50)

**TEXT EFFECTS**

* Conditional text: [?(have X)|yes|no]
* Data embedding: [this], [thingKey:ps], [n:thingKey]
* Tags: <b>, <i>, <q>, <#f00>, <//>, <.>

**DESIGN RECOMMENDATIONS**

* Study Bunny Clicker for formatting standards.
* Use templates for reusability.
* Use icons and flavor text to boost clarity.
* Style with CSS or define stylesheet:.
* Back up your source files.
* Never rename declared keys (breaks saves).

**TESTING AND SHARING**

* Test locally using hosted .txt URL.
* Share using: https://orteil.dashnet.org/igm/?g=yourgameurl
* Join r/idlegamemaker or Discord for support.

**USEFUL EXAMPLES**

* Bunny Clicker source: <https://orteil.dashnet.org/igm/games/bunnyclicker.txt>
* Try using includes or templates for bulk upgrade handling.

**FINAL REMARKS** Idle Game Maker allows deep customization and progression systems through a combination of structured syntax and creative logic. “Bunny Clicker” is an ideal reference for designing polished, fun, and interactive idle games.

Created by Orteil and Opti. Engine by DashNet.