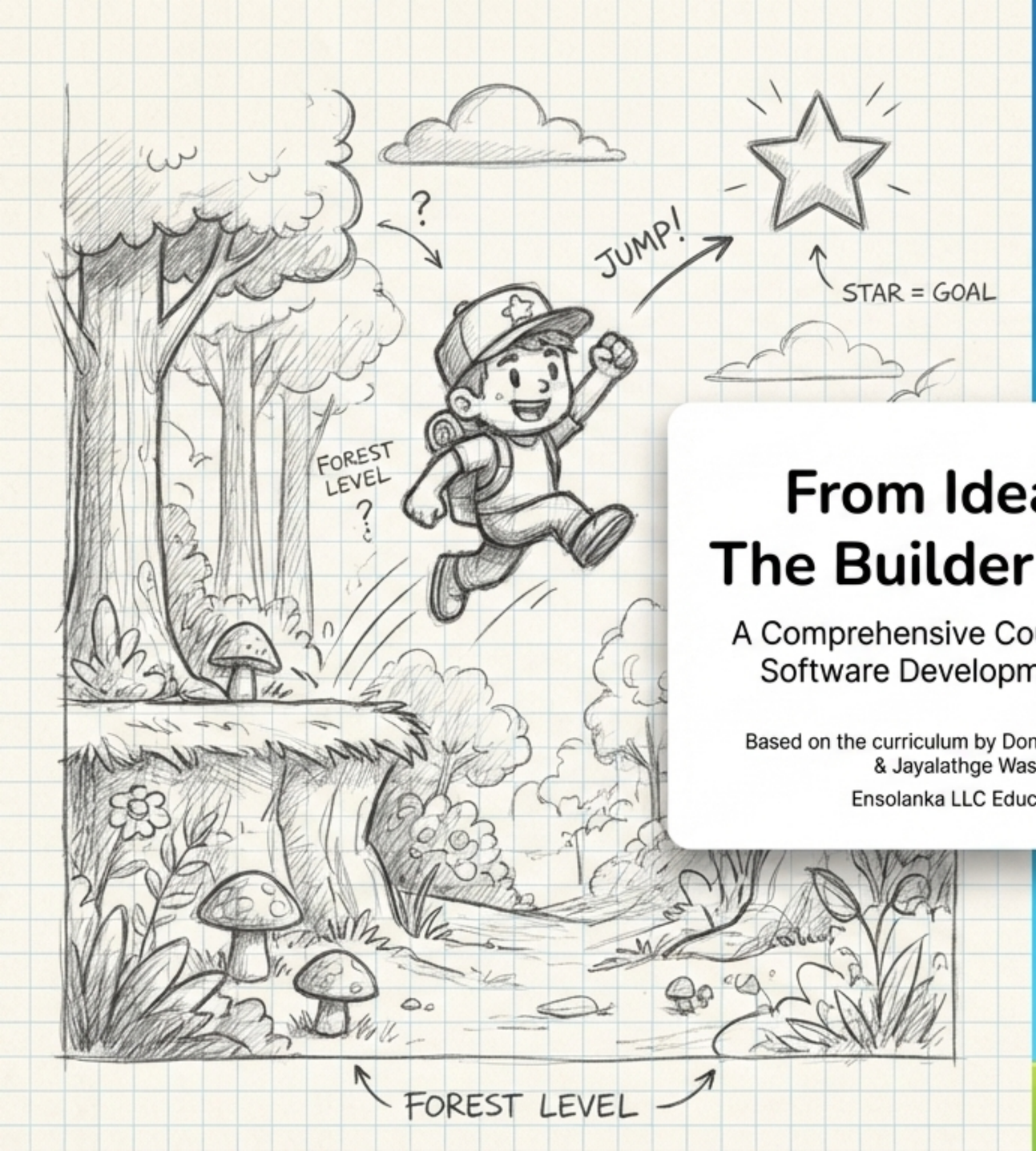


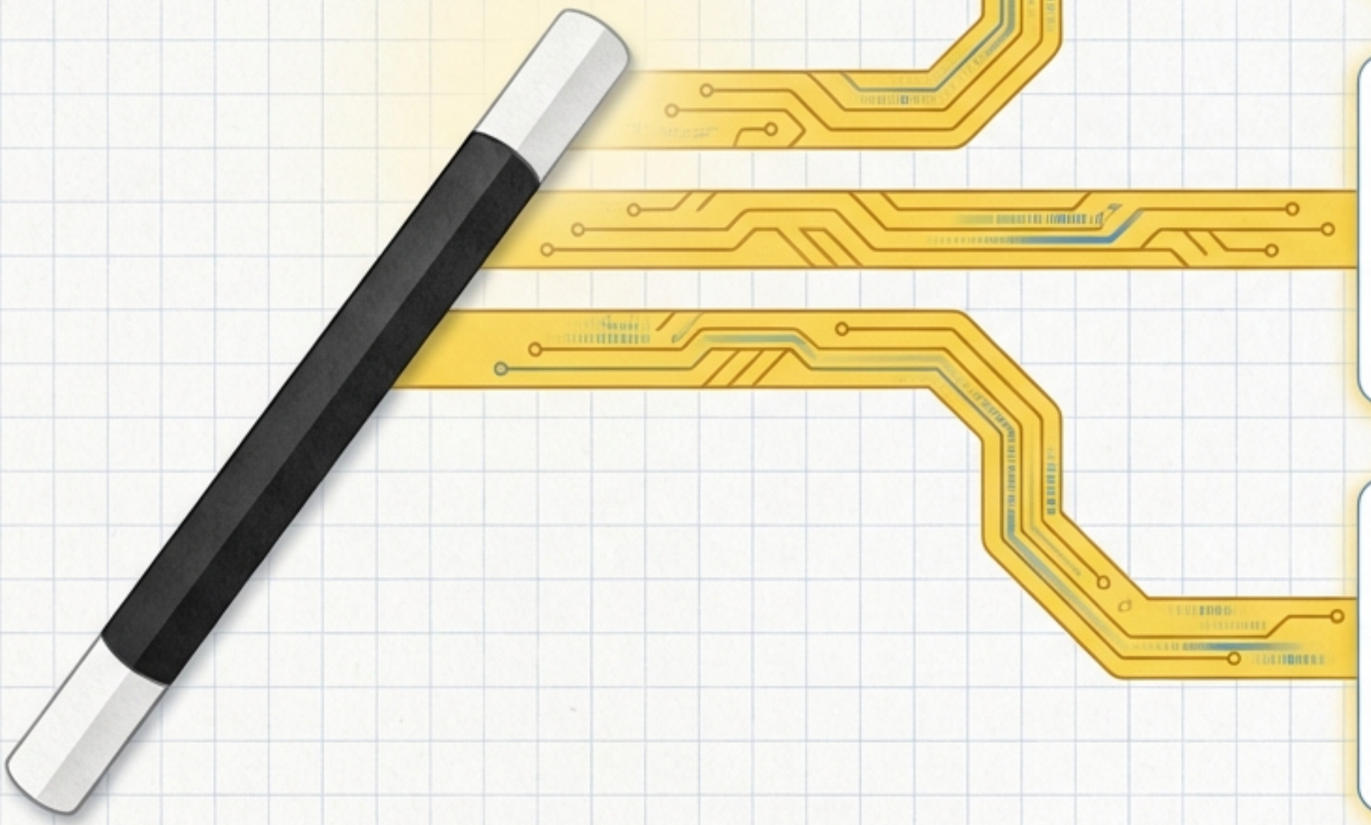
From Idea to App: The Builder's Blueprint

A Comprehensive Course Overview of the Software Development Cycle for Kids.

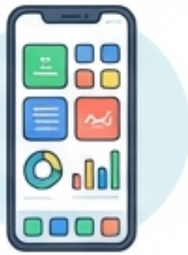
Based on the curriculum by Don Sugath Wasantha Jayathunge & Jayalathge Wasana Randeepani.

Ensolanka LLC Educational Framework.

Software Development is the Modern Magic Wand



Turn passive tech consumers into active digital creators.

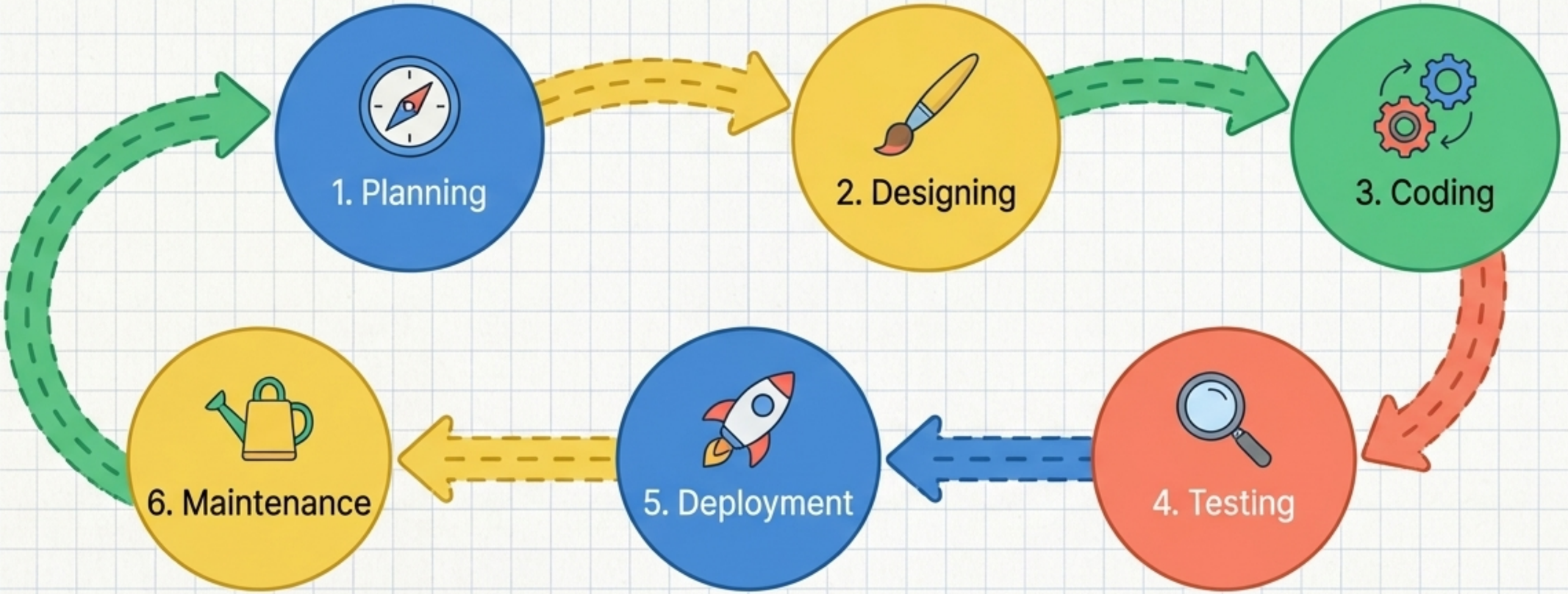


Learn the exact professional steps developers use to build the tools that change the world.

Software development is the process of turning ideas into something you can use on a computer, tablet, or phone.

The Six Levels of Software Creation

Creating software is a lot like baking a cake—you follow a recipe, mix the ingredients, bake it, and then enjoy the delicious result. Master the complete cycle from the first spark of an idea to a live, working application.



Planning: Breaking Big Ideas into Actionable Blueprints

Idea Generation

Every great software project starts with a cool idea.

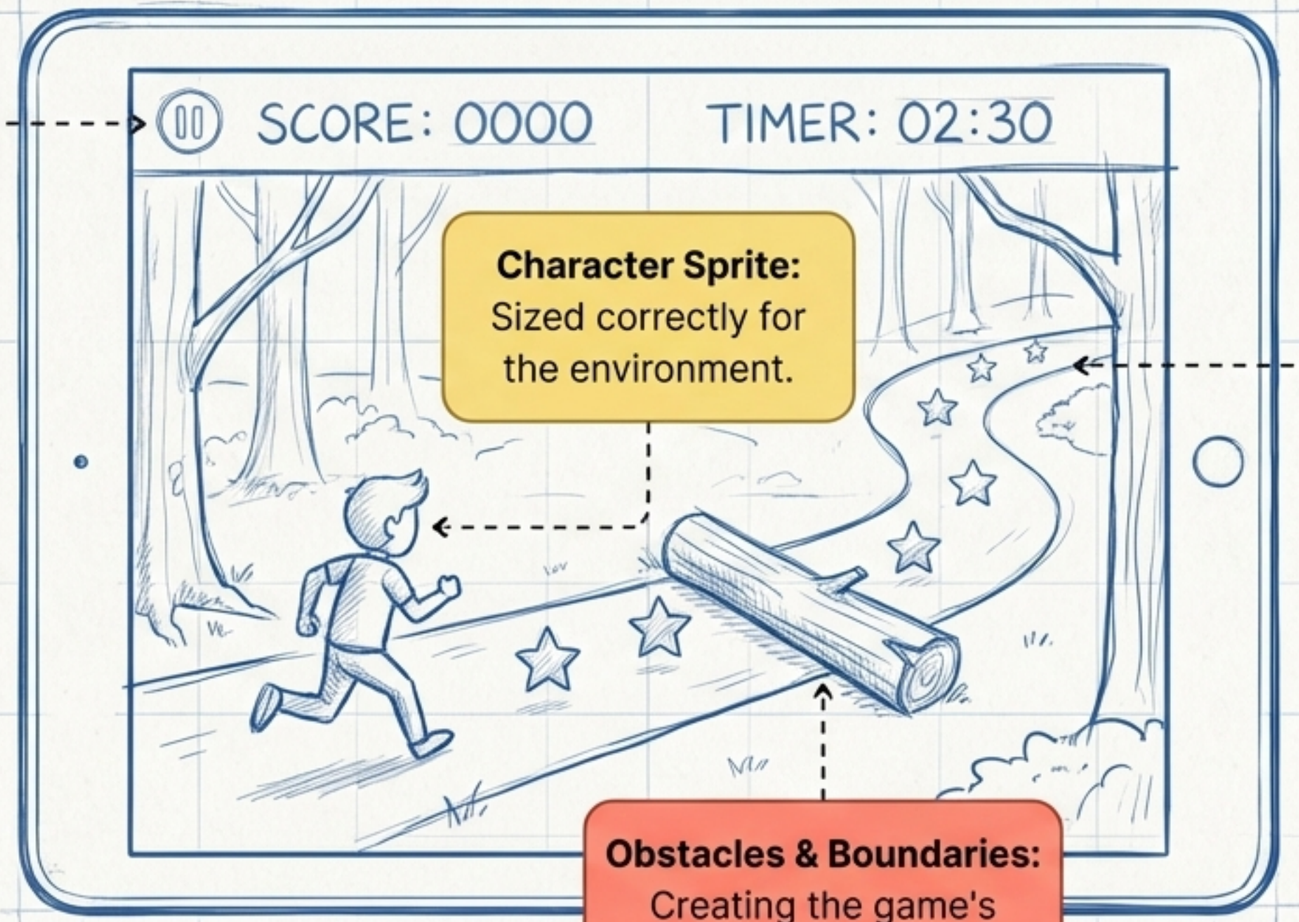
Task Breakdown

Students learn to visualize their end goal and divide the treasure map into bite-sized, manageable coding tasks.

Designing: Sketching the Digital Canvas

Before coding, we sketch. Designing is where ideas start to take shape and look like the final product by mapping out the user journey.

Score & Timer:
Keeping track of player progress.



Character Sprite:
Sized correctly for the environment.

Collectibles (Stars):
Placed at varying difficulties.

Obstacles & Boundaries:
Creating the game's physical rules.

The Developer’s Toolkit: Languages that Grow with the Student

Choosing the right tool for the job. The curriculum supports platforms that teach fundamental logic first, seamlessly scaling to real-world syntax.

Scratch

- **Type:** Visual, block-based.
- **Advantage:** Snap together code; instant visual feedback.
- Perfect for absolute beginners.

Python

- **Type:** Text-based.
- **Advantage:** Simple, powerful, and widely used by professional developers.
- Transitions students to real-world engineering.

Coding: Giving Ideas a Voice and an Engine

Coding is where we turn those ideas into something that actually works. Students write instructions to build windows, apply physics, and trigger collision detection.

Step 1

Workshop Logic: **Set the Stage** (Window & Colors)

```
screen = pygame.display.set_mode((800, 600))
```

Step 2

Workshop Logic: **Spawn the Hero** (Draw the character)

```
pygame.draw.rect(screen, character_color,
[character_x, character_y...])
```

Step 3

Workshop Logic: **Add Physics (Arrow key movement)**

```
if keys[pygame.K_RIGHT]:
  character_x += character_speed
```

Step 4

Workshop Logic: **The Goal (Collision detection)**

```
if character_x < star_x < character_x +
character_width:
```

Testing: The Ultimate Quality Control

Why We Test: Like making sure your spaceship is ready for takeoff before blasting off. The method is deliberately trying to break the game to find glitches.

Self-Testing

Play through all levels. Try everything. Move in all directions, jump over obstacles, and take notes on edge cases.

Peer Testing

Get fresh eyes on the game. Gather feedback on the user experience and brainstorm fun new ideas for improvement.

The Bug Hunter’s Guide: Becoming a Code Detective

Debugging is like being a detective, hunting down bugs in your code and solving them so your game works perfectly.

Syntax Errors

Symptom: Mistyped words or missing brackets (e.g., forgotten parenthesis).

The Fix: Read the error line, check for missing characters or commas.

Logic Errors

Symptom: Code runs, but behaves wrong (e.g., character moves left instead of right).

The Fix: Review command order and ensure your logic conditions are correct.

Runtime Errors

Symptom: Game crashes mid-play (e.g., off-screen asset or dividing by zero).

The Fix: Track the crash trigger and write code to handle impossible edge cases.

Deployment: Opening the Doors to the Digital World

The Milestone: It’s the moment when all your hard work pays off, and others get to enjoy what you’ve made.

1 Prepare Files

2 Create Account

3 Write Description

4 Hit Publish!

Maintenance: Tending to the Living Application

The Reality: Software is never truly “done.” Students learn the professional responsibility of monitoring performance and responding to player feedback over time.

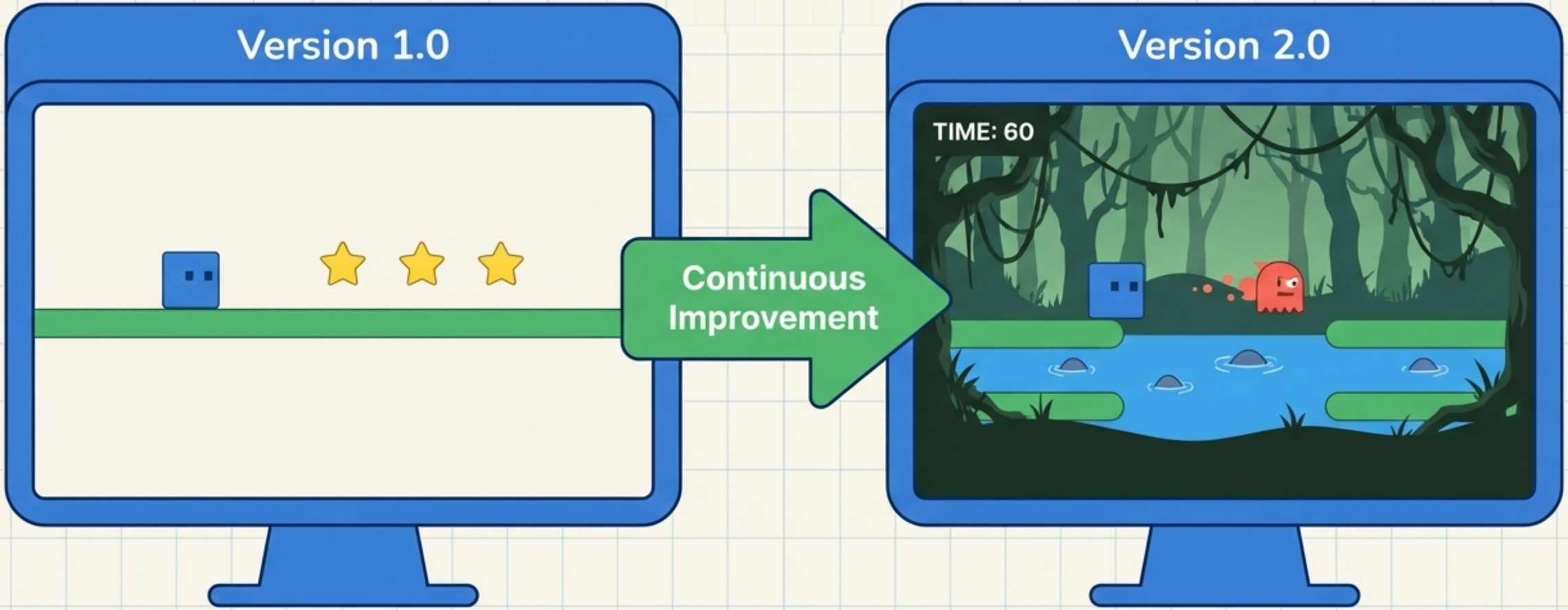
Watering: Adding new features and levels to keep the game fresh and exciting.

Pruning: Fixing newly discovered bugs and glitches as more people play.

Sunlight: Updating for modern compatibility and optimizing overall game performance.

The Iteration Cycle: Leveling Up the Software

Adding new features is like growing your garden... it keeps things fresh! Expanding Version 1.0 into Version 2.0 teaches students how to code new mechanics and increase difficulty.



Core Vocabulary: Fluency in the Coder's Lexicon

Equipping students with the professional terminology required to navigate the modern tech landscape.

Algorithm:
A set of instructions or rules designed to solve a problem.

Variable:
A container in your code that holds data, like numbers or text.

Loop:
A sequence of instructions that repeats until a condition is met.

Function:
A reusable block of code that performs performs a specific task.

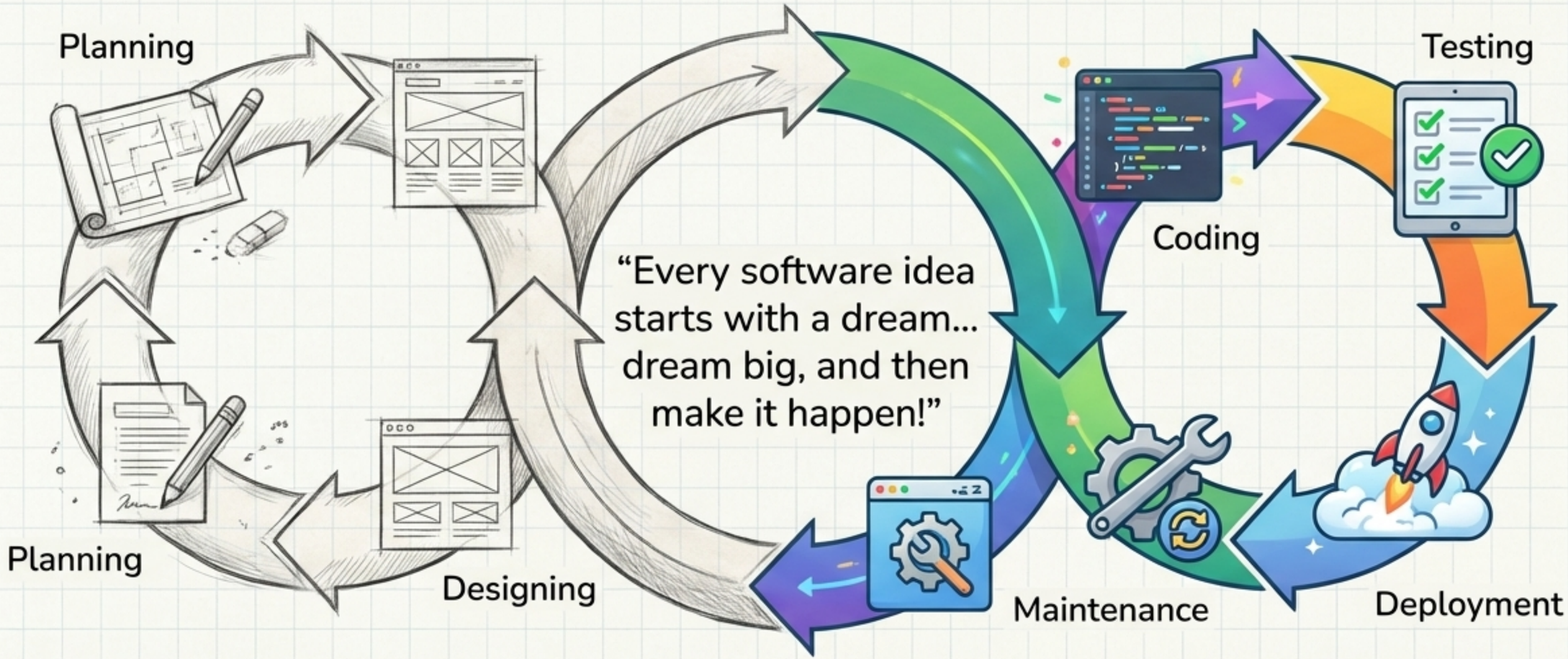
Bug:
An error or flaw in your code that causes unexpected behavior.

Interface:
The part of the software that the user actually interacts with.

Beyond the Course: Blueprints for the Future

The skills you've learned and the projects you've created are the foundation for a lifetime of exploration.

Turn Screen Time into Build Time



Empower the next generation of digital creators. Enroll Today.