

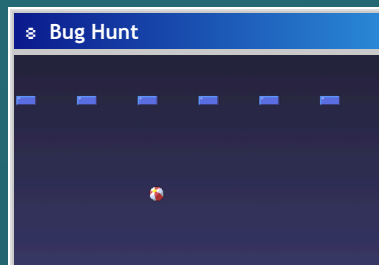
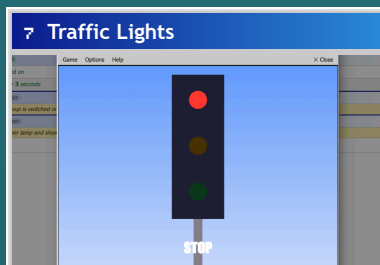
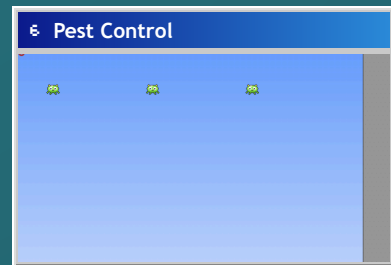
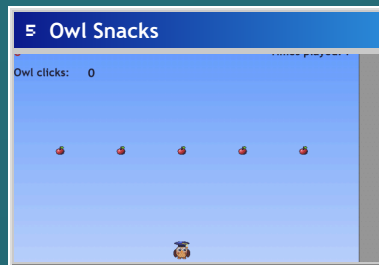
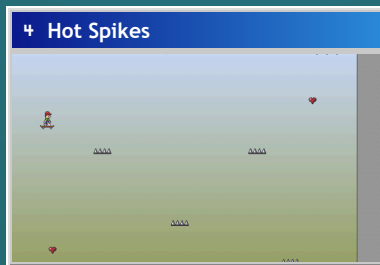
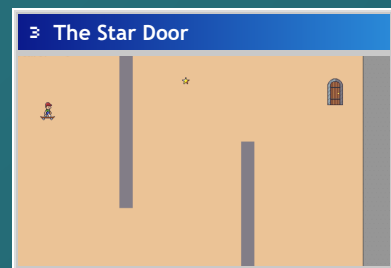
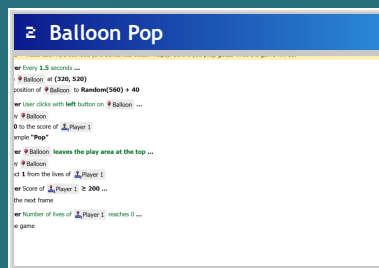
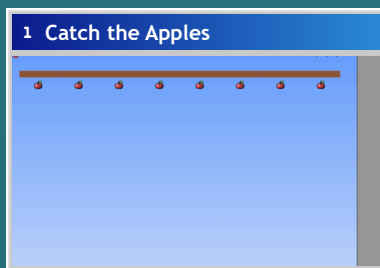


LESSON PLANS

Whenever

in the Classroom

Nine lessons of about 50 minutes that teach the logic behind programming, using Whenever, a game maker where rules are ticked in a grid rather than typed. Each lesson has a ready-made starter game in **File** ▶ **Open a lesson starter**.



About these lessons

The course at a glance

L1 Cause and effect

An event: whenever something happens, do something.

L2 Reading rules aloud

Rules are sentences; you can predict what a program will do by reading it.

L3 And

Every condition in an event must be true at the same time.

L4 Moments and states

Something that happens once is different from something that stays true.

L5 Remembering things

Variables: named places to keep a number.

L6 Which one?

An action affects only the copies its conditions picked.

L7 Modes

A system is always in one mode, and rules switch between modes.

L8 Finding bugs

Debug by comparing what you expected with what happened.

L9 Make it yours

Plan, build, test and share a game for someone else.

Assessment

Handouts

Rule cards · Moment or state? · Bug log · Design brief

undefined



About these lessons

The lessons suit pupils of about 9 to 14 who have not programmed before. Each takes 45 to 60 minutes and builds on the one before. Lessons 1 to 4 are the core; lessons 5 to 9 go further.

Pupils meet the big ideas of programming without typing any code: events, conditions joined with *and*, moments and states, variables, choosing which object an action affects, modes, debugging, and designing for other people. Every rule they write reads as a sentence, so they can say their program out loud.

How a lesson runs

Part	Time	What happens
Warm-up	5–10 min	An unplugged game or a question that introduces the idea.
Demonstration	5–10 min	You show the key move on the board or projector, thinking aloud.
Activity	20–30 min	Pupils work in the lesson's starter game, in pairs.
Stretch	As time allows	Extra tasks for pairs who finish early.
Check	5 min	One exit question that shows whether the idea has landed.

Each lesson below gives objectives, key words, the steps, the mix-ups to watch for, and an answer key: the events a finished game needs.

Before the first lesson

- **Computers:** Whenever runs in a web browser (Chrome, Edge, Firefox or Safari), with nothing to install. Building games needs a mouse or trackpad with a right-click; tablets are fine for playing the finished games.
- **Where it lives:** put one copy on your school website or any static web host. The User's Guide explains how, in *Putting Whenever on a website*.
- **Starter games:** pupils open them from **File** ► **Open a lesson starter**. You can also hand out a link ending in `?lesson=` and the lesson number, such as `.../whenever/?lesson=3`, which opens lesson 3's starter straight away.

- **Saving:** each browser keeps one game automatically. On shared computers that is not enough, so pupils should save a file at the end of every lesson (**File ▶ Save game**) and load it next time (**File ▶ Load game...**). A name like *Sam lesson 3* helps.
- **Sound:** headphones help. A running game's **Options** menu can switch sounds off.
- **Pairs:** pair programming works well. One pupil drives the mouse; the other navigates, reading the rules aloud. Swap every ten minutes.

Every editor is explained in detail in Whenever User's Guide. Its chapter "Your first game" builds a whole game step by step, which makes a good first session for you before teaching lesson 1.

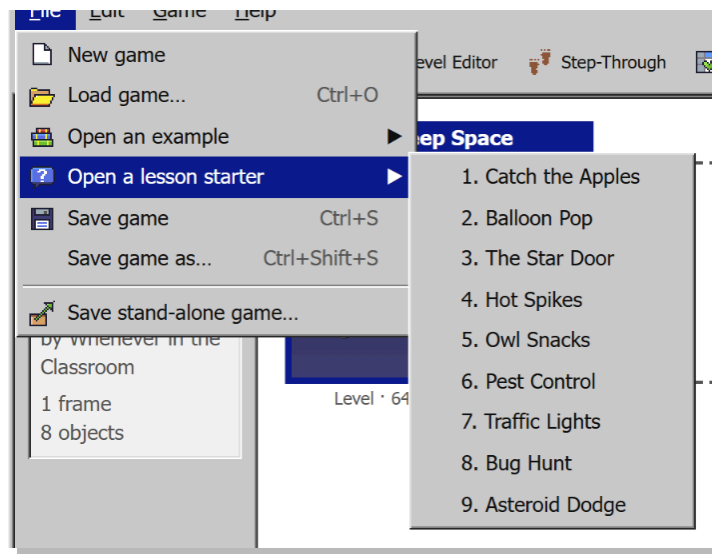


FIG 1 The File menu, with the nine lesson starters

■ Teaching tips

- **Say rules as sentences:** **Whenever** the ball collides with a brick, destroy the brick. The Event Editor's **Sentences** button shows them that way.
- **Predict, then play:** before running a change, ask what will happen. Then check.
- **Ask "which event did that?"** before "how do I fix it?" It points pupils at the cause.
- **One change at a time:** change one thing, play, and only then change the next.

undefined



The course at a glance

#	Lesson	Big idea	Starter game	What pupils use
1	Cause and effect	An event: whenever something happens, do something.	Catch the Apples	Step-Through Editor
2	Reading rules aloud	Rules are sentences; you can predict what a program will do by reading it.	Balloon Pop	Sentence view
3	And	Every condition in an event must be true at the same time.	The Star Door	Event Editor, a counter
4	Moments and states	Something that happens once is different from something that stays true.	Hot Spikes	Green and black conditions, limit conditions
5	Remembering things	Variables: named places to keep a number.	Owl Snacks	Counters, alterable values, global values
6	Which one?	An action affects only the copies its conditions picked.	Pest Control	Picking at random, families
7	Modes	A system is always in one mode, and rules switch between modes.	Traffic Lights	Groups of events
8	Finding bugs	Debug by comparing what you expected with what happened.	Bug Hunt	The debugger
9	Make it yours	Plan, build, test and share a game for someone else.	Asteroid Dodge	Stand-alone games

Lesson 1

1

Cause and effect

Pupils learn that a program is a set of rules of the form *whenever something happens, do something*. They write every rule of a small game by playing it in the Step-Through Editor.

STARTER GAME

Catch the Apples. Apples hang from a branch and drop one at a time, and the bat follows the mouse. Nothing happens yet when an apple is caught or missed. One ready-made rule drops the apples; pupils don't need to change it.

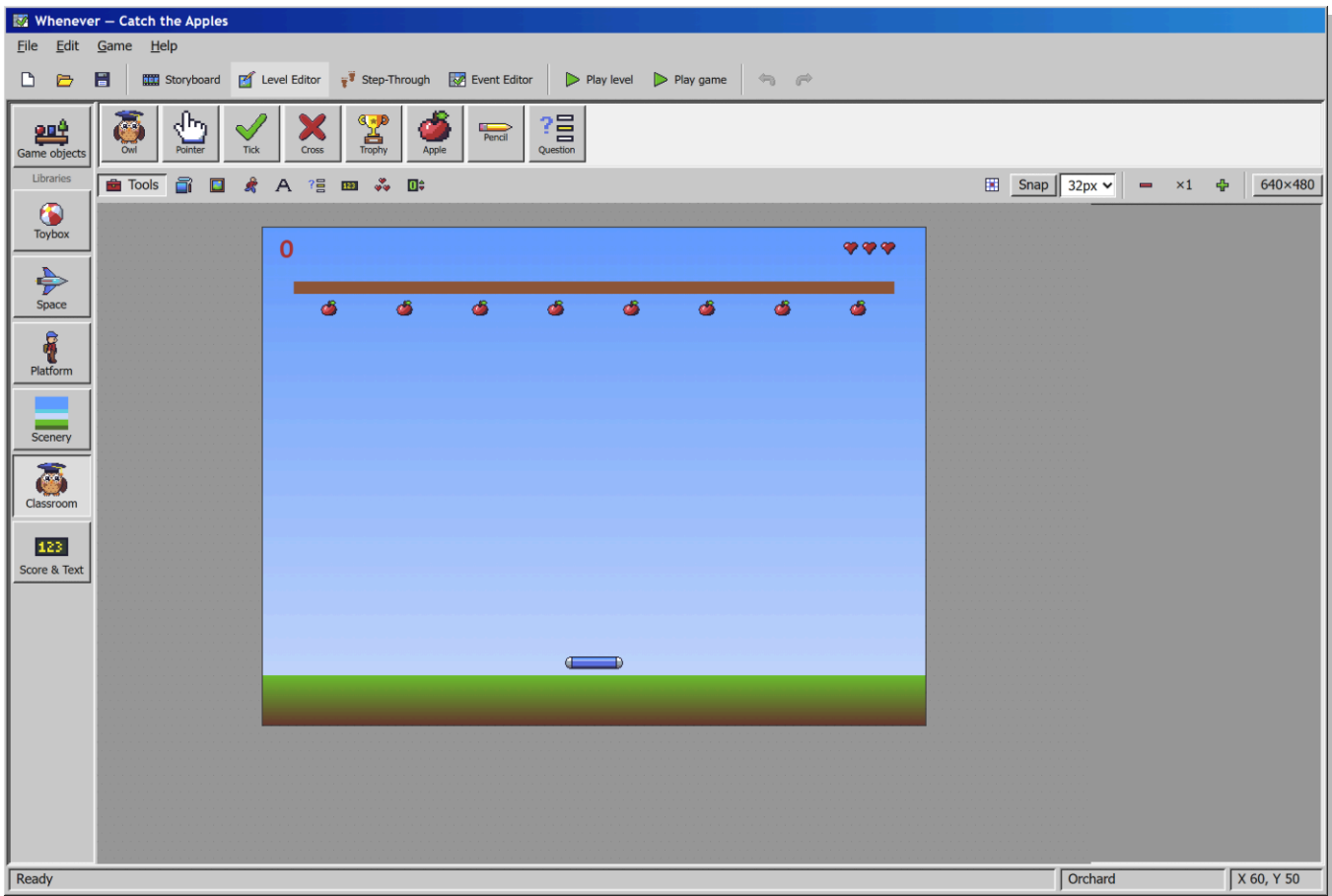


FIG 1.1 Catch the Apples in the Level Editor: apples on a branch, the bat above the grass

■ Objectives

- ✓ Explain an event as *whenever something happens, do something*.
- ✓ Add events with the Step-Through Editor while playing.
- ✓ Test a game, and change a rule that doesn't feel right.

Key words:

event

condition

action

test

■ Warm-up: classroom rules 10 MIN

Give the class a few rules and play them: **Whenever** *I clap, stand up. Whenever I say a colour, touch your head. Whenever I clap twice, sit down.* Then ask pupils to invent one. Point out the two halves of every rule: the thing that happens (the **condition**) and what you do (the **action**).

■ Demonstration 5 MIN

Open the starter and click **Step-Through**. When the game stops on *Apple leaves the play area at the bottom*, think aloud: "What should happen when we miss an apple?" Click the **Player 1** icon and choose **Number of lives** ▶ **Subtract from number of lives**, with 1. Click the **Apple** icon and choose **Destroy**. Click **Continue game**.

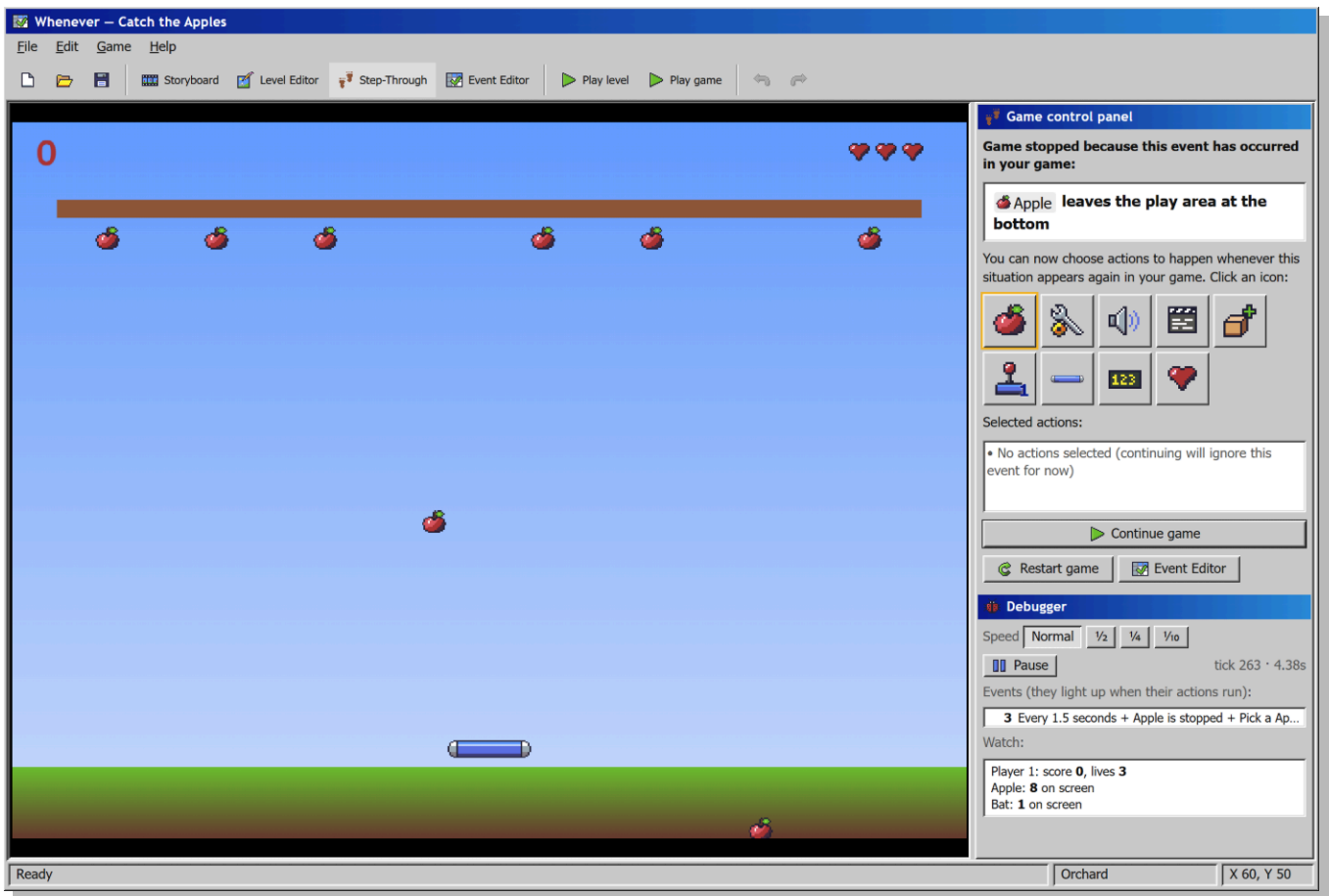


FIG 1.2 The Step-Through Editor stopped because an apple fell off the bottom

Activity 25 MIN

In pairs:

- 1 Open the starter and click **Step-Through**.
- 2 Each time the game stops, read the message aloud, agree what should happen, and choose the actions. Then click **Continue game**.
- 3 Make catching an apple worth a point: when the bat touches an apple, destroy the apple and add 1 to the score.
- 4 Make a missed apple cost a life.
- 5 Open the **Event Editor** and click **Sentences**. Read your rules aloud to each other.

Stretch

- Play a sound when an apple is caught: **Sound** ► **Play sample**, then pick one from the sound maker.
- Instead of destroying a missed apple, put it back: **Position** ► **Set Y coordinate...** to 78, then **Movement** ► **Stop**. The ready-made rule will drop it again, so the apples never run out.

- End the game when the lives run out.

■ Check for understanding

"Write one rule from your game as a sentence that starts with **Whenever**." Look for both halves: a condition, and at least one action.

■ Common mix-ups

- ✗ **Continuing without choosing an action.** The game then ignores that situation for the rest of the run. Click **Restart game** to be asked again.
- ✗ **Acting on the wrong object**, such as destroying the Bat. The objects involved are outlined in the panel; check whose icon you clicked.
- ✗ **"Bat collides with Apple" or "Apple collides with Bat"?** They mean the same thing, so either is fine.

■ Answer key

	Condition	Actions
1	● Every 1.5 seconds ● Apple is stopped ● Pick an Apple at random (ready-made)	✓ Start Apple
2	● Apple leaves the play area at the bottom	✓ Subtract 1 from the lives of Player 1 ✓ Destroy Apple
3	● Bat collides with Apple	✓ Destroy Apple ✓ Add 1 to the score of Player 1
4	● Number of lives of Player 1 reaches 0 (stretch)	✓ End the game

Lesson 2



Reading rules aloud

Pupils read a finished game's rules, predict what the game will do, and then play to check. Reading code before running it is a skill programmers use every day.

STARTER GAME

Balloon Pop. A complete game in five rules. Balloons float up from the bottom; click them to pop them before they escape.

The screenshot shows the 'Whenever - Balloon Pop' game editor. The interface includes a menu bar (File, Edit, Game, Help), a toolbar with icons for Storyboard, Level Editor, Step-Through, Event Editor, Play level, and Play game, and a status bar at the bottom showing 'Ready', 'Party', and coordinates 'X 60, Y 50'. The main area displays the 'Sentences' view of the game rules, which are listed as follows:

- Lesson 2 — Read each rule out loud (the Sentences button helps). Before you play, guess what the game will do.
- Whenever** Every 1.5 seconds ...
 - Create Balloon at (320, 520)
 - Set X position of Balloon to $\text{Random}(560) + 40$
- Whenever** User clicks with left button on Balloon ...
 - Destroy Balloon
 - Add 10 to the score of Player 1
 - Play sample "Pop"
- Whenever** Balloon leaves the play area at the top ...
 - Destroy Balloon
 - Subtract 1 from the lives of Player 1
- Whenever** Score of Player 1 ≥ 200 ...
 - Go to the next frame
- Whenever** Number of lives of Player 1 reaches 0 ...
 - End the game

FIG 2.1 Balloon Pop's five rules in the sentence view

Objectives

- ✓ Read an event aloud as a sentence.
- ✓ Predict what a program will do from its rules, and test the prediction.
- ✓ Change one rule and predict the effect of the change.

Key words: rule predict random timer

Warm-up: rule cards 10 MIN

Before pupils see the game, give each group these five cards. Ask: "What kind of game is this? Draw what the screen looks like."

Card	Rule
A	Every 1.5 seconds, make a new balloon just below the bottom of the screen, at a random place from left to right.
B	Whenever someone clicks a balloon, destroy that balloon, add 10 to the score, and play a pop.
C	Whenever a balloon floats off the top, destroy it and take away a life.
D	Whenever the score is 200 or more, go to the next screen.
E	Whenever the lives reach 0, end the game.

Demonstration 5 MIN

Open the starter, go to the Event Editor and click **Sentences**. Read the first rule aloud and match it to card A. Ask what `Random(560) + 40` might mean. (A random number from 40 to 599: the balloon's distance from the left edge.)

Activity 25 MIN

1 Predict. Before playing, each pair writes answers to these questions:

- 1 How often does a new balloon appear?
- 2 How many balloons must you pop to win?
- 3 If you never click anything, how many balloons escape before the game ends?
- 4 What happens to a balloon you click?

2 Test. Play, and mark each prediction right or wrong.

3 Change and predict again. Make one change at a time, predict, then play: change *Every 1.5 seconds* to 0.5; change the 10 points to 50; disable rule C (right-click its number, **Disable event**).

■ Stretch

Write a new rule as a sentence first, then build it: **Whenever** the score of Player 1 is 100 or more, play the Pop sound. Predict, then play. It sounds like a buzz, because the sound starts again sixty times a second. Lesson 4 explains why.

■ Check for understanding

Show a rule the class hasn't seen, such as **Whenever** the user clicks a balloon, add 1 to the lives of Player 1. Ask what it would change about the game.

■ Common mix-ups

- ✗ **"Every 1.5 seconds" is a clock, not a count.** It doesn't mean "1.5 balloons".
- ✗ **Rule order doesn't matter much here.** The game checks every rule many times a second, so pupils shouldn't read the list as steps done once from top to bottom.
- ✗ **A balloon that is created and a balloon that is placed** behave the same: rules apply to every balloon.

■ Answer key

Question	Answer
How often does a new balloon appear?	Every 1.5 seconds.
How many balloons to win?	20, because 200 points ÷ 10 points each.
Never click: how many escape?	3, one for each life. Then the game ends.
What happens to a clicked balloon?	It is destroyed, the score goes up by 10, and a pop plays.

Lesson 3



And

Pupils build an event with two conditions and learn that both must be true at the same time. A door opens only when the skater touches it *and* has found all three stars.

STARTER GAME

The Star Door. A small maze with three stars, a counter that counts them, and a door. Collecting stars already works; the door does nothing yet.

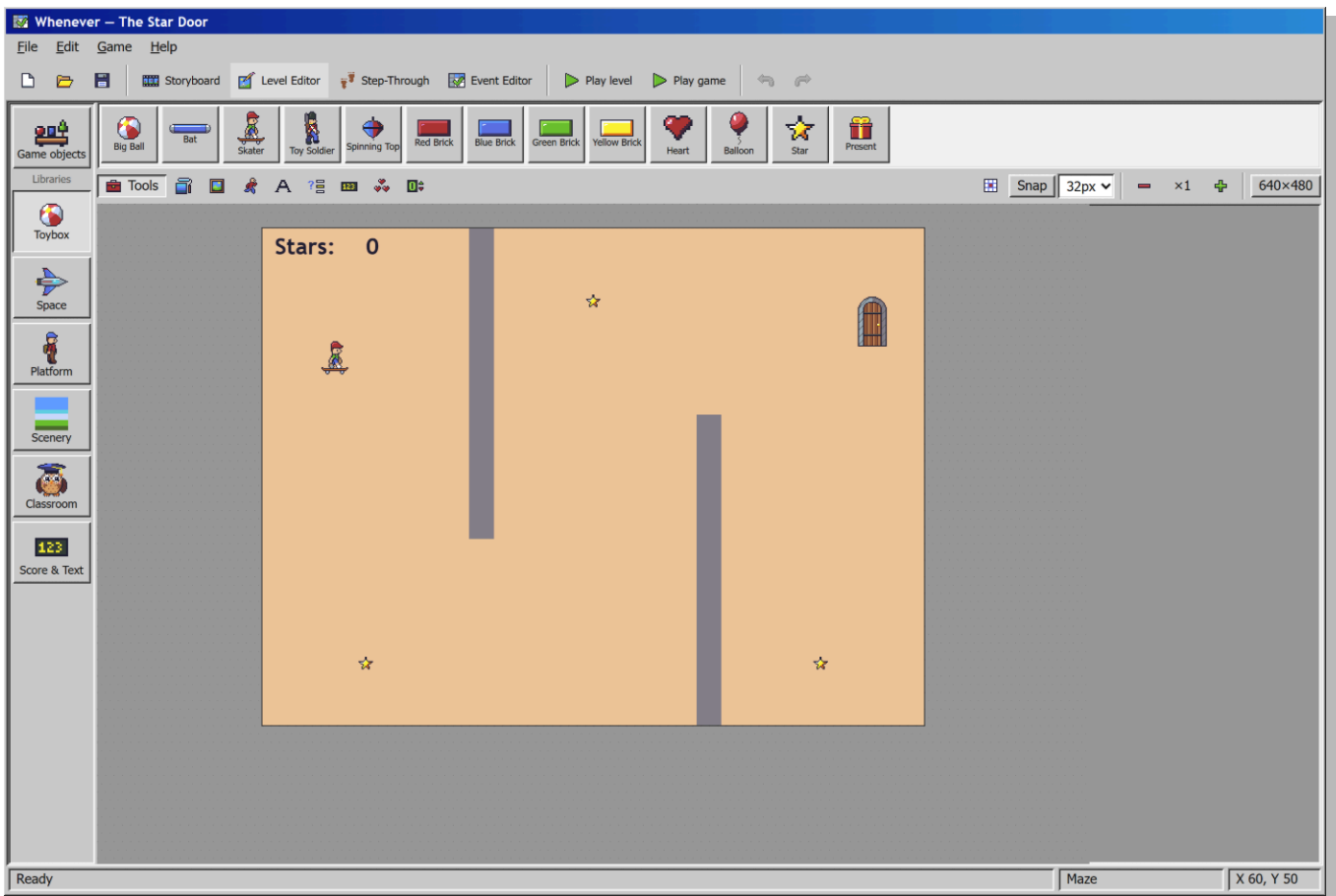


FIG 3.1 The Star Door: the skater, two walls, three stars, the door and the Stars counter

Objectives

- ✓ Add a second condition to an event.
- ✓ Explain that every condition in an event must be true together.
- ✓ Test a condition with a counter.

Key words: condition and counter true false

Warm-up: stand up if... 10 MIN

"Stand up if you are wearing something blue." Then: "Stand up if you are wearing something blue **and** you have a brother or sister." Who stood up the first time but not the second? Then try **or**, and compare. (In Whenever, *or* is two separate events.)

Demonstration 5 MIN

Open the Event Editor. Read the star event aloud: **Whenever** the Skater collides with a Star, destroy the Star and add 1 to Stars. Play, collect a star, and watch the counter. Then show how to give an event a second condition: right-click its number and choose **Add a new condition**.

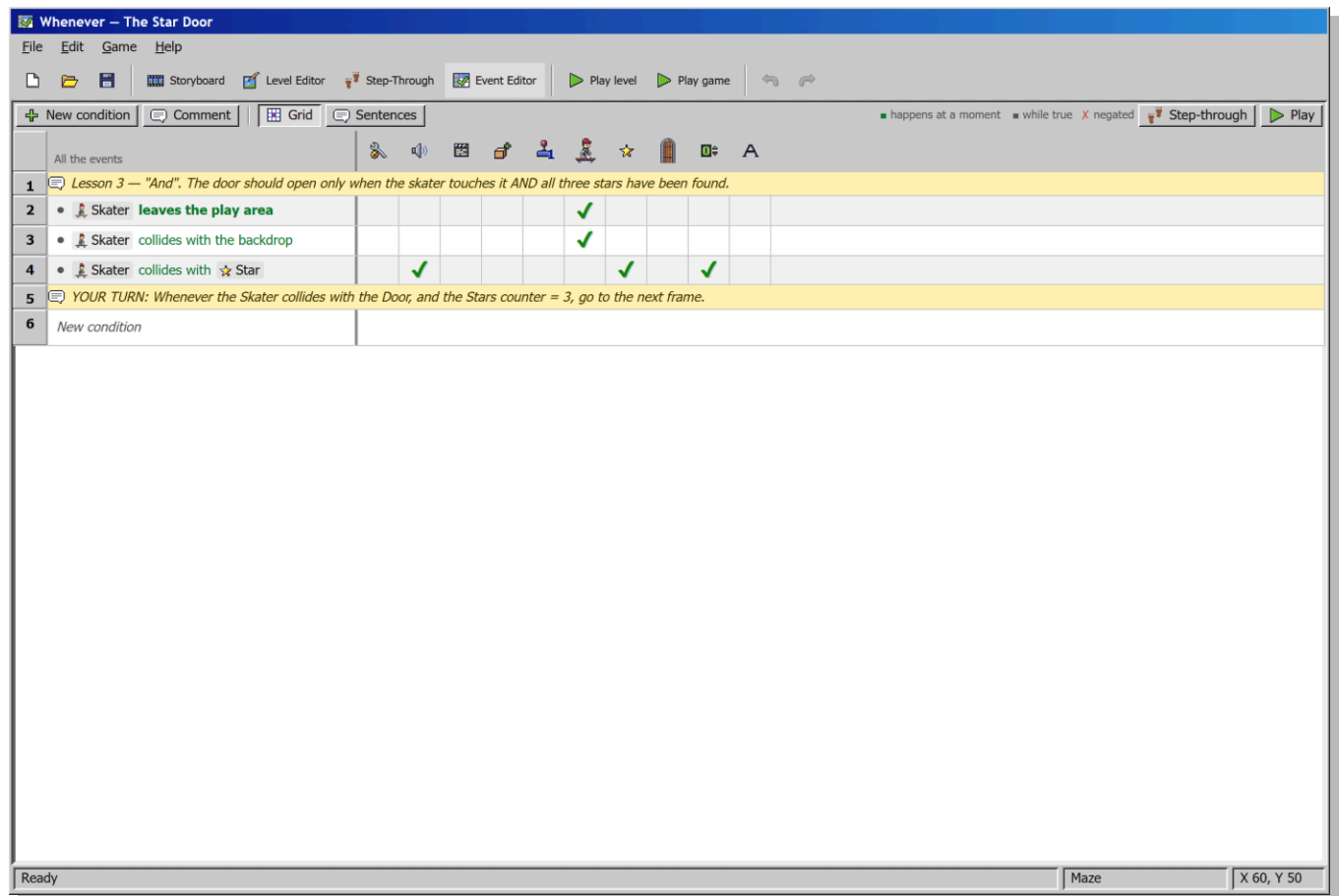


FIG 3.2 The starter's events, with the task written as a comment

Activity 25 MIN

- 1 Play: collect the stars, then touch the door. Nothing happens.
- 2 Click **New condition**, click **Skater**, choose **Collisions** ▶ **Another object**, and pick the **Door**.
- 3 Right-click the new event's number and choose **Add a new condition**. Click the **Stars** counter, choose **Compare the counter to a value**, and set it to = 3.
- 4 In the same row, right-click the box under **Storyboard controls** (the clapperboard) and choose **Next frame**.
- 5 Test it both ways: touch the door with two stars, then with three.

Stretch

- **A locked-door message.** Make a Text object with **Tools** (*Find all three stars first!*, ticked **Invisible at start**). Add an event: the Skater collides with the Door **and** Stars < 3 → **Display text...**
- **Negate.** Make the same message a different way: copy the door event, right-click its Stars = 3 condition and choose **Negate**.
- **A trap.** Add Spikes. Whenever the Skater collides with Spikes **and** Stars > 0, subtract 1 from Stars.

Check for understanding

Ask pupils to fill in the last column.

Touching the door?	Stars	Does the door open?
Yes	2	X No
No	3	X No
Yes	3	✓ Yes

Common mix-ups

- X **Two events instead of one.** A separate *Skater collides with Door* event opens the door with no stars at all. Both conditions must be in the same row.
- X **New condition instead of Add a new condition.** The toolbar's **New condition** starts a new event; the right-click menu adds to an existing one.
- X **The wrong comparison.** Stars = 3 works; Stars > 3 never comes true with three stars. Stars ≥ 3 also works.

■ Answer key

	Conditions	Actions
1	● Skater collides with Star (ready-made)	✓ Destroy Star ✓ Add 1 to Stars ✓ Play sample Ding
2	● Skater collides with Door ● Stars = 3	✓ Go to the next frame
3	● Skater collides with Door ● Stars < 3 (stretch)	✓ Display the message text



STARTER GAME

Whenever — Hot Spikes

File Edit Game Help

New Storyboard Level Editor Step-Through Event Editor Play level Play game Undo Redo

+ New condition Comment Grid Sentences happens at a moment while true X negated Step-through Play

All the events	🔍	🔊	📺	🗑️	🧑	🚶	⏸️	💖	❤️	⌛	➡️	A	A
Lesson 4 — Moments and states. Green conditions happen at a moment. Black ones stay true, and their actions repeat sixty times a second.													
Skater leaves the play area						✓							
Hold the Space bar down for a second. Why do these two counters count so differently?													
Repeat while Space bar is pressed										✓			
Upon pressing Space bar											✓		
BUG: touching a spike takes ALL your lives at once. Why? Can you fix it in three different ways?													
Skater is overlapping Spikes		✓				✓							
Skater collides with Heart						✓		✓					
Number of lives of Player 1 reaches 0			✓										
New condition													

Ready Spike Room X 60, Y 50

FIG 4.1 Hot Spikes' events: green moments, black states, and the buggy spike event

■ Objectives

- ✓ Tell a moment condition (green) from a state condition (black).
- ✓ Explain why actions repeat while a black condition is true.
- ✓ Fix a repeating action in more than one way, and choose the best fix.

Key words: moment state while tick repeat

■ Warm-up: clap when, clap while 5 MIN

"Clap **when** my hand goes up." (One clap.) "Clap **while** my hand is up." (Clapping until it comes down.) Same hand, same clap, different rule.

■ Demonstration 5 MIN

Play the starter and hold the Space bar down for a second. **Held** races up by about sixty a second; **Presses** goes up by one. Show the two events: *Repeat while Space bar is pressed* is black, *Upon pressing Space bar* is green.

■ Activity 30 MIN

- 1 Play and touch a spike. All three lives vanish at once and the game ends.
- 2 Find the spike event in the Event Editor. Is its condition green or black? Why does that explain the bug?
- 3 Fix it three ways. After each fix, play, then **Undo** (Ctrl+Z) before trying the next.
 - 1 Right-click the condition, choose **Replace...**, and pick **Skater** ▶ **Collisions** ▶ **Another object** ▶ **Spikes** (green).
 - 2 Keep *is overlapping*, right-click the event's number, **Add a new condition**, and choose **Special** ▶ **Limit conditions** ▶ **Only one action when event loops**.
 - 3 Keep *is overlapping* and add **Special** ▶ **Limit conditions** ▶ **Restrict actions for...** 1 second.
- 4 Discuss: which fix makes the best game? Fixes 1 and 2 take one life per touch. Fix 3 takes one life a second while you stand on a spike.

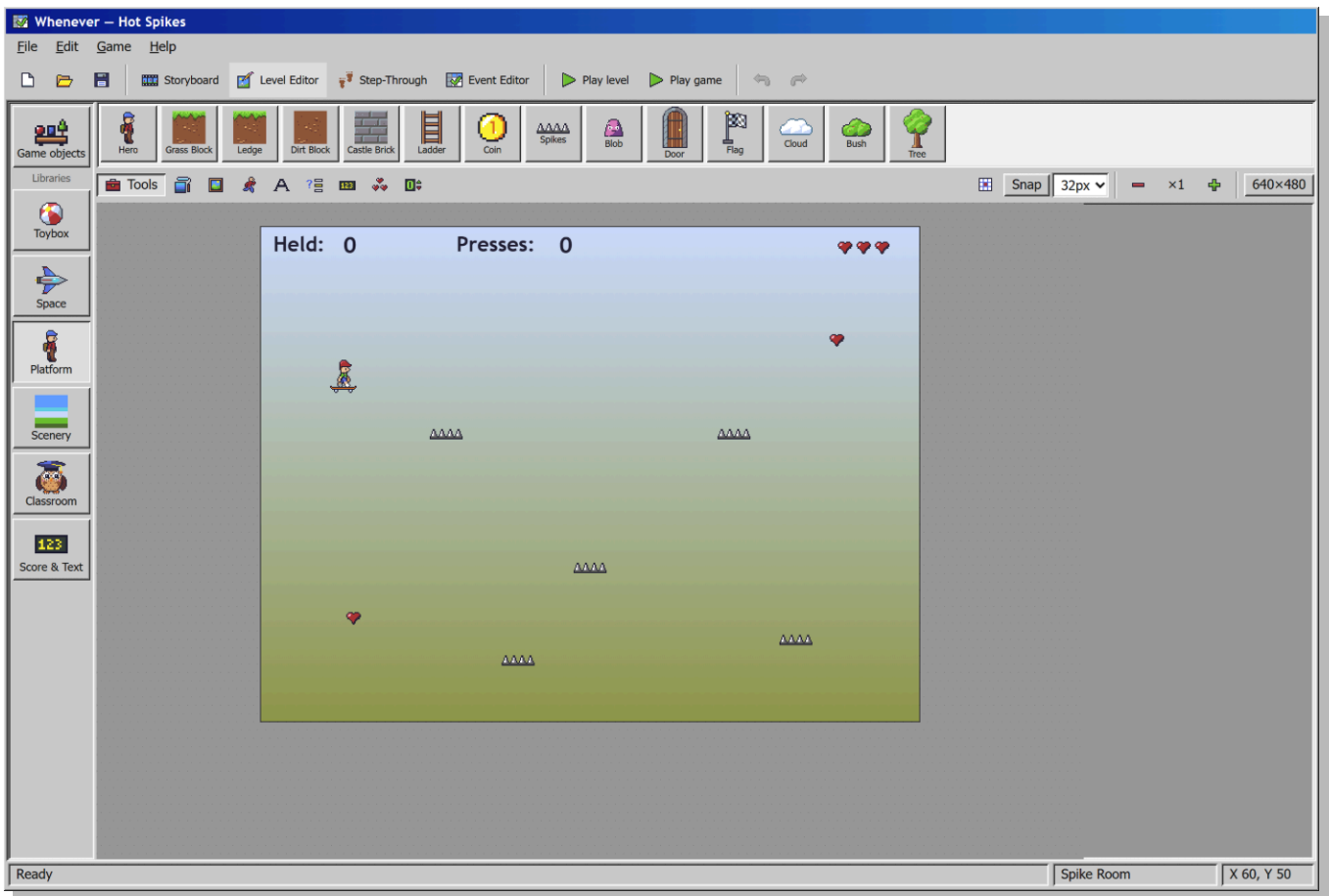


FIG 4.2 The Hot Spikes room: five spikes, two hearts and the skater

Stretch

Use a state on purpose. Add a Text object and the event *Always* → *Set text to a value...* `"Held for " + Round(Counter("Held") / 60) + " seconds"`. *Always* is black, so the text updates every tick, which is exactly what we want.

Check for understanding

Sort these into **moment** and **state**: *collides with*, *is overlapping*, *upon pressing a key*, *score of Player 1 > 100*, *every 2 seconds*, *repeat while a key is pressed*.

Common mix-ups

- ✗ **"Is overlapping" sounds like the moment of touching.** It means "is touching right now", which stays true for many ticks.
- ✗ **"The computer does it once."** Pupils often picture the game reading the rules once. It reads them sixty times a second.

X States aren't bad. They are right for things that should keep happening, like a score display or a slow drain while standing in lava.

■ Answer key

Fix	Condition	How it feels
1	Skater collides with Spikes	One life per touch.
2	Skater is overlapping Spikes + Only one action when event loops	One life per touch.
3	Skater is overlapping Spikes + Restrict actions for 1 second	One life a second while touching: a chance to escape.

The card sort: moments are *collides with*, *upon pressing a key* and *every 2 seconds*; states are *is overlapping*, *score > 100* and *repeat while a key is pressed*.



Remembering things

Pupils meet variables: named places where a program keeps a number. Whenever has four kinds, and choosing the right one is the real skill.

Kind	Belongs to	Example in Owl Snacks	Where to find it
Score and lives	A player	The score	Player 1 icon
Counter	One object on the screen	Owl clicks	A Counter object (Score & Text library)
Alterable value	Each copy of an object	Each apple's Bites	The object's Values menu
Global value	The whole game, and can be saved on the computer	Times played	Special ► Global values

STARTER GAME

Owl Snacks. Five apples and a hungry owl. Every apple has its own alterable value called *Bites*, starting at 3. Clicking the owl already adds 1 to a counter; the three tasks are written as comments in the Event Editor.

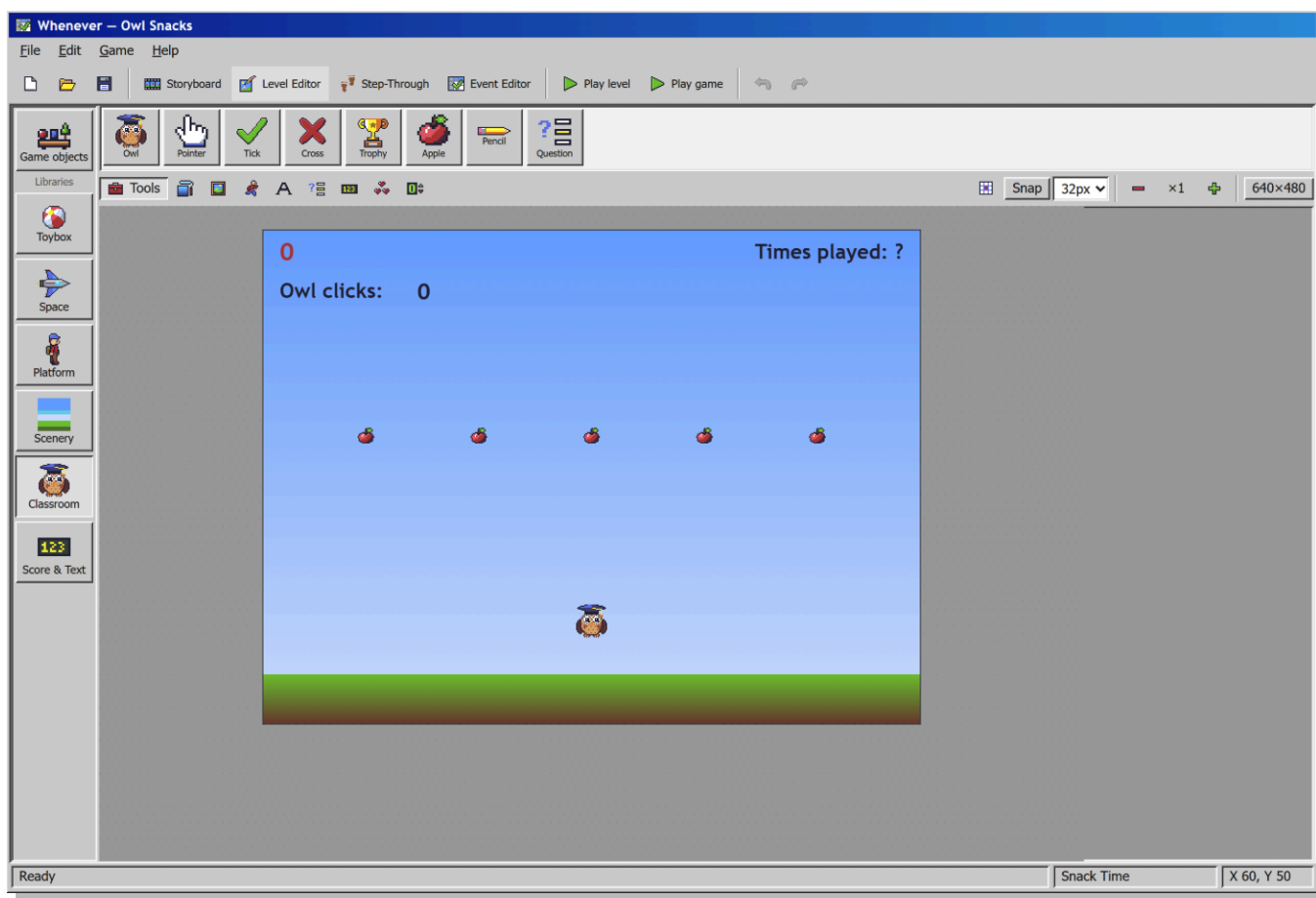


FIG 5.1 Owl Snacks: five apples, the owl, the score, the Clicks counter and the Times played text

■ Objectives

- ✓ Describe a variable as a named place to keep a number that can change.
- ✓ Choose between a counter, an alterable value and a global value.
- ✓ Change a variable with an action and test it with a condition.

Key words:

variable

value

counter

alterable value

global value

save

■ Warm-up: sticky notes 10 MIN

Give five pupils a sticky note saying *Bites*: 3. Each time you "bite" one (tap their shoulder), they cross out the number and write one less. At 0, they sit down. Ask: why does each apple need its own note? What would go wrong with one note for all five?

■ Demonstration 5 MIN

Play the starter and click the owl a few times; the Clicks counter goes up. Show the event that does it. Then right-click an apple, choose **Alterable values...**, and show *Bites*, starting at 3.

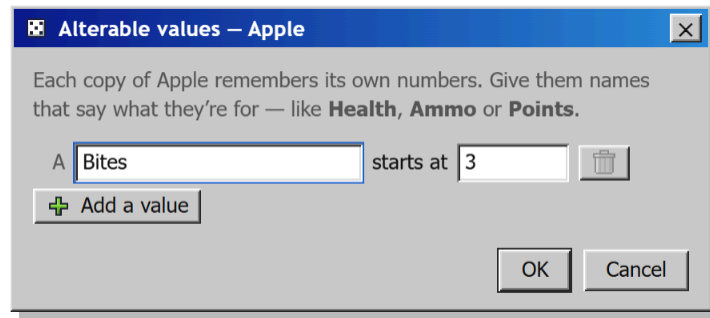


FIG 5.2 The Alterable values box for the Apple: Bites starts at 3

Activity 25 MIN

- 1 **Task 1: take a bite.** Click **New condition**, then **Mouse & keyboard** ▶ **The mouse** ▶ **User clicks on an object**, and choose the Apple. In that row, right-click the box under **Apple** and choose **Values** ▶ **Subtract from value: Bites, 1**. Add **Sound** ▶ **Play sample** ▶ **Crunch**.
- 2 **Task 2: eat it.** New condition: **Apple** ▶ **Values** ▶ **Compare to an alterable value, Bites = 0**. Actions: **Destroy** the Apple; Player 1 **Score** ▶ **Add to score, 10**.
- 3 **Test.** Click one apple twice and another once. Each remembers its own bites.
- 4 **Task 3: remember between games.** New condition: **Storyboard controls** ▶ **Start of frame**. Under **Special**, add in this order: **Global values** ▶ **Load from this computer** (Played), **Add to a global value** (Played, 1), **Save on this computer** (Played). Under **Played Text**, add **Set text to a value...** with `"Times played: " + Global("Played")`.
- 5 Close the game, play it again, and watch the number. It survives even after the browser is closed.

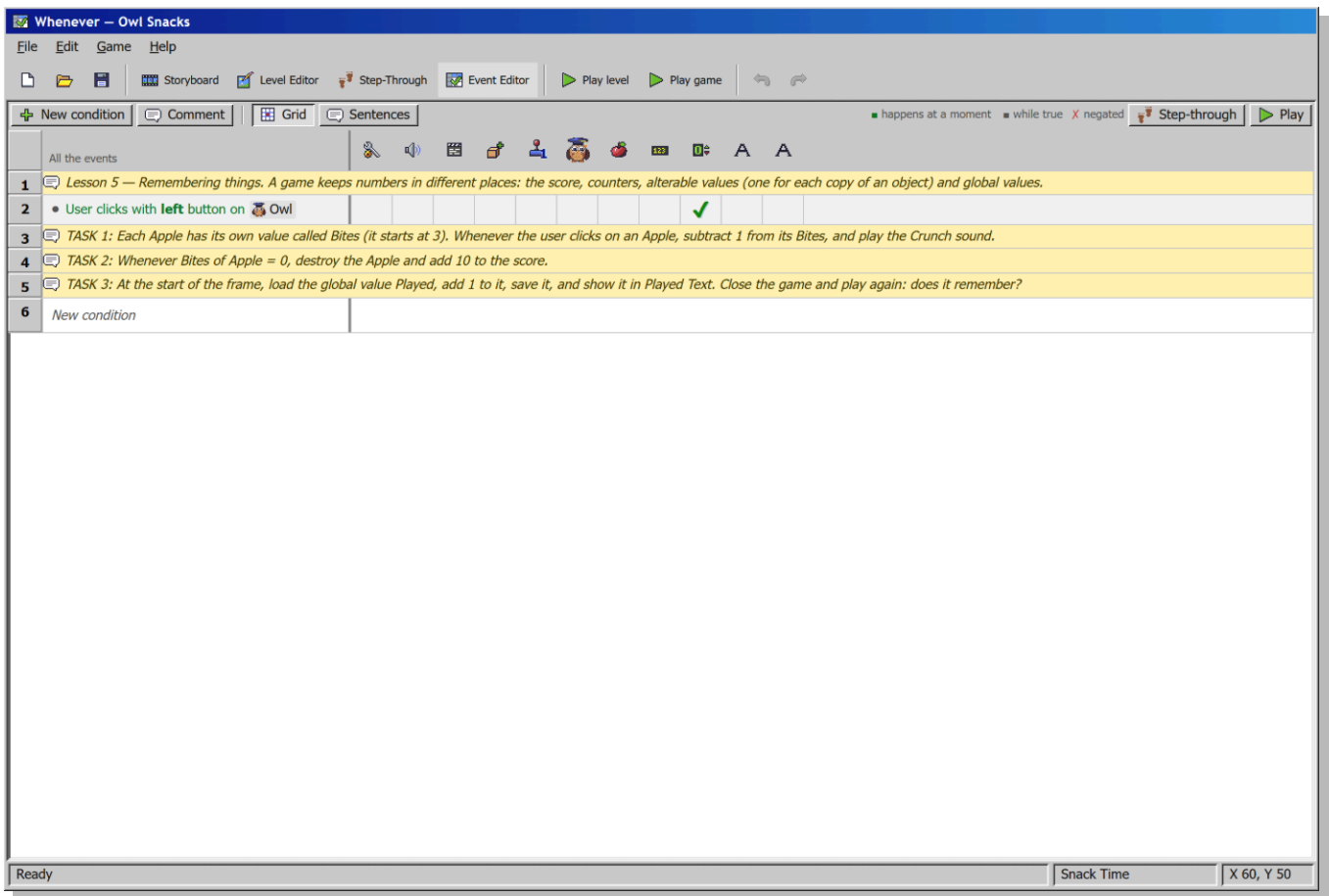


FIG 5.3 The starter's events: the owl counter, then the three tasks as comments

Stretch

- **Random apples.** At the start of the frame, set each apple's Bites to `Random(4) + 1`, so some take one bite and some take four.
- **Thank you.** When all the apples have been destroyed, show a Text object that says *Thank you!*

Check for understanding

"The owl's clicks are a counter, but the bites are alterable values. Why couldn't one counter remember the bites for all five apples?"

Common mix-ups

- ✗ **User clicks, not User clicks on an object.** The plain *User clicks* condition doesn't pick an apple, so every apple loses a bite at once. Lesson 6 explains why.
- ✗ **Order in Task 3.** Load, add, save, then show. Saving before adding, or forgetting to load, starts the count at 1 every time.

X Shared computers. Saved global values belong to the browser, so another pupil's games count too.

■ Answer key

	Condition	Actions
1	● User clicks with left button on Owl (ready-made)	✓ Add 1 to Clicks
2	● User clicks with left button on Apple	✓ Subtract 1 from Bites of Apple ✓ Play sample Crunch
3	● Bites of Apple = 0	✓ Destroy Apple ✓ Add 10 to the score of Player 1
4	● Start of frame	✓ Load Played ✓ Add 1 to Played ✓ Save Played ✓ Set the text of Played Text to "Times played: " + <code>Global("Played")</code>

Lesson 6



Which one?

Pupils learn that conditions **pick** objects, and actions act only on the copies that were picked. When no condition picks an object, an action on it affects every copy. They also group different objects into a family so one rule covers them all.

STARTER GAME

Pest Control. Four blobs patrol the grass and three aliens march across the sky. There are no rules yet: just four tasks, written as comments.

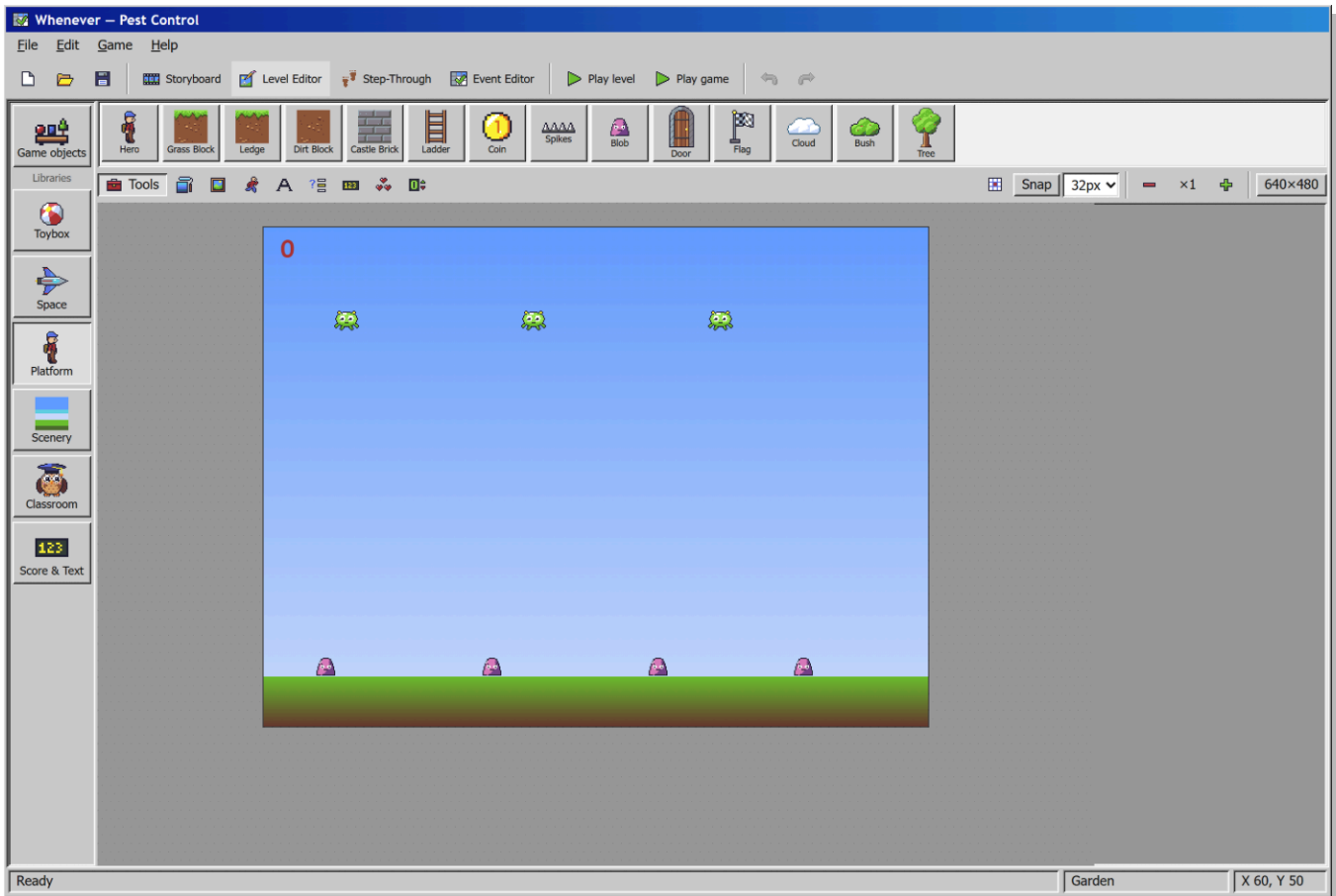


FIG 6.1 Pest Control: four blobs on the grass, three aliens in the sky

Objectives

- ✓ Predict which copies of an object an action will affect.
- ✓ Pick one copy at random.
- ✓ Put different objects in a family and write one rule for the whole family.

Key words: copy pick random family

Warm-up: who waves? 5 MIN

"Everyone wearing glasses, wave." Only some wave: the condition picked them. "Wave!" Everyone waves: nothing picked anyone, so it means all of you. Then draw one name from a hat: picking at random.

Demonstration 5 MIN

Do Task 1 together: **New condition** ▶ **Mouse & keyboard** ▶ **The mouse** ▶ **User clicks on an object** ▶ **Blob**, then **Destroy** under Blob. Play and click a blob. Ask: "Why did only that one disappear?"

Activity 25 MIN

- Task 1.** As in the demonstration, and add 10 to the score.
- Task 2: the surprise.** In the same row, add **Destroy** under **Alien**. Predict, then play and click a blob. Every alien vanishes. Discuss why: the condition picked a blob, but nothing picked an alien, so *Destroy Alien* means every alien. Undo it afterwards.
- Task 3: pick at random.** New condition: **Timer** ▶ **Every...** 2 seconds. Right-click the event's number, **Add a new condition: Blob** ▶ **Pick or count** ▶ **Pick one at random**. Action: **Blob** ▶ **Movement** ▶ **Reverse**. Play: one blob turns round every 2 seconds.
- Task 4: a family.** Right-click a blob, choose **Families...**, type *Pests* and click **Add**. Right-click an alien, choose **Families...** and tick *Pests*. Now write one event: user clicks on **any Pests** → **Destroy** any Pests, add 10 to the score, play the Splat sound. Delete the old blob-only event.

Stretch

- **Win.** Whenever **any Pests** ▶ **Pick or count** ▶ **Compare to the number of these objects** is 0, end the game.
- **Breeding pests.** Every 3 seconds, pick a blob at random and **Create new objects** ▶ **Create object:** an Alien, *relative to* the Blob. The new alien appears at the picked blob.

■ Check for understanding

Show two rules. *Hero collides with Coin* → *Destroy Coin*: how many coins disappear? (One.) *Hero collides with Coin* → *Destroy Star*: how many stars? (All of them.)

■ Common mix-ups

- ✗ **Task 2 is meant to surprise.** Let pupils explain it before you do.
- ✗ **Picking in the wrong order.** Put *Pick one at random* after the conditions that narrow things down, so it picks from what is left.
- ✗ **A family is a label, not a new object.** The blobs and aliens keep their own pictures and movements.

■ Answer key

	Conditions	Actions
1	● User clicks with left button on Blob	✓ Destroy Blob ✓ Add 10 to the score of Player 1
2	● Every 2 seconds ● Pick a Blob at random	✓ Reverse Blob
3	● User clicks with left button on any Pests	✓ Destroy any Pests ✓ Add 10 to the score ✓ Play sample Splat
4	● Number of any Pests = 0 (stretch)	✓ End the game

Lesson 7



Modes

Pupils learn that many systems are always in exactly one **mode** (or state), and that rules decide when to move from one mode to the next. In Whenever, each mode is a **group** of events, and only one group is switched on at a time. Computer scientists call this a state machine.

STARTER GAME

Traffic Lights. A traffic light with three groups: Red, Green and Amber. The Red group is finished as an example. Green and Amber are empty, apart from a comment saying what to build.

The screenshot shows the 'Whenever - Traffic Lights' software interface. The top menu bar includes 'File', 'Edit', 'Game', and 'Help'. Below the menu is a toolbar with icons for 'New condition', 'Comment', 'Grid', 'Sentences', and various event types. The main workspace is divided into sections for 'All the events', 'Red', 'Green', and 'Amber' groups.

Red Group (2 events):

- 1. Lesson 7 — Modes. A traffic light is always in one mode: red, green or amber. Each mode is a group of events, and only one group is switched on at a time.
- 2. When this group is switched on
- 3. This group has been on for 3 seconds

Green Group (1 event):

- 4. YOUR TURN: when this group is switched on, light the green lamp (dim the others) and show GO. After 3 seconds, switch to Amber.

Amber Group (1 event):

- 5. YOUR TURN: light the amber lamp and show GET READY. After 1 second, switch back to Red.

New condition (1 event):

- 6. New condition

The bottom status bar shows 'Ready', 'Crossroads', and 'X 60, Y 50'.

FIG 7.1 The Traffic Lights events: the finished Red group, and the empty Green and Amber groups

Objectives

- ✓ Describe a system that is always in one of several modes.
- ✓ Use a group of events for each mode.
- ✓ Switch between modes with actions, and draw the modes as a state diagram.

Key words: mode state group switch on switch off transition

Warm-up: be the traffic light 10 MIN

Draw the light's modes on the board as circles, with arrows between them, and the time spent in each. Three pupils each hold a coloured card; only one card may be up at a time. Call out the seconds and let them run the cycle.

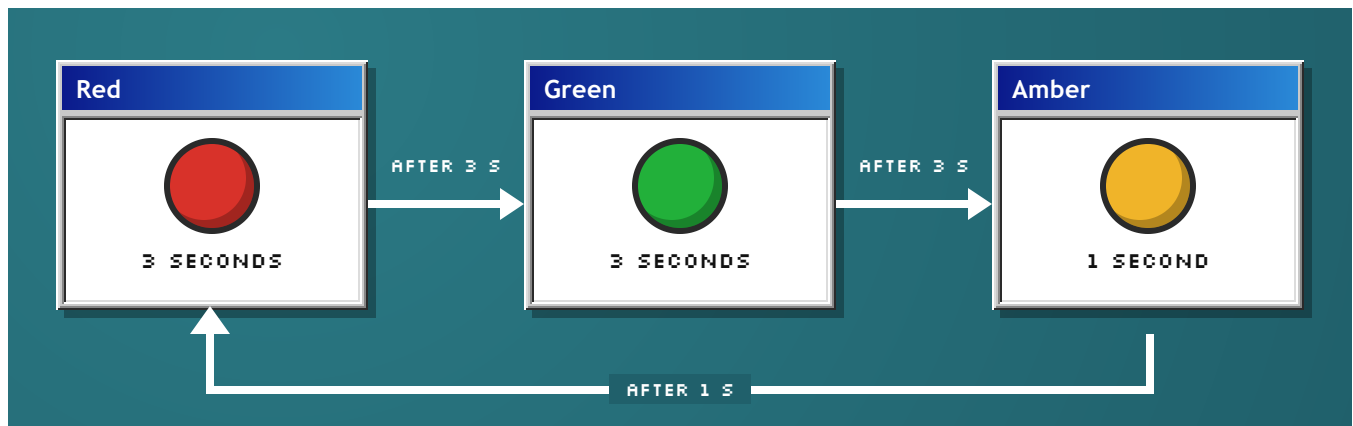


DIAGRAM Traffic Lights · three modes, one at a time

Each box is a group of events. Each arrow is an *After this group has been on for...* event that switches its own group off and the next one on.

Demonstration 5 MIN

Play the starter: the light goes red and the sign says STOP. Read the Red group aloud: *When this group is switched on, light the red lamp, dim the others and show STOP. When this group has been on for 3 seconds, switch Red off and switch Green on.* After 3 seconds the light stays red. Ask why. (Green is switched on, but it has no events yet.)

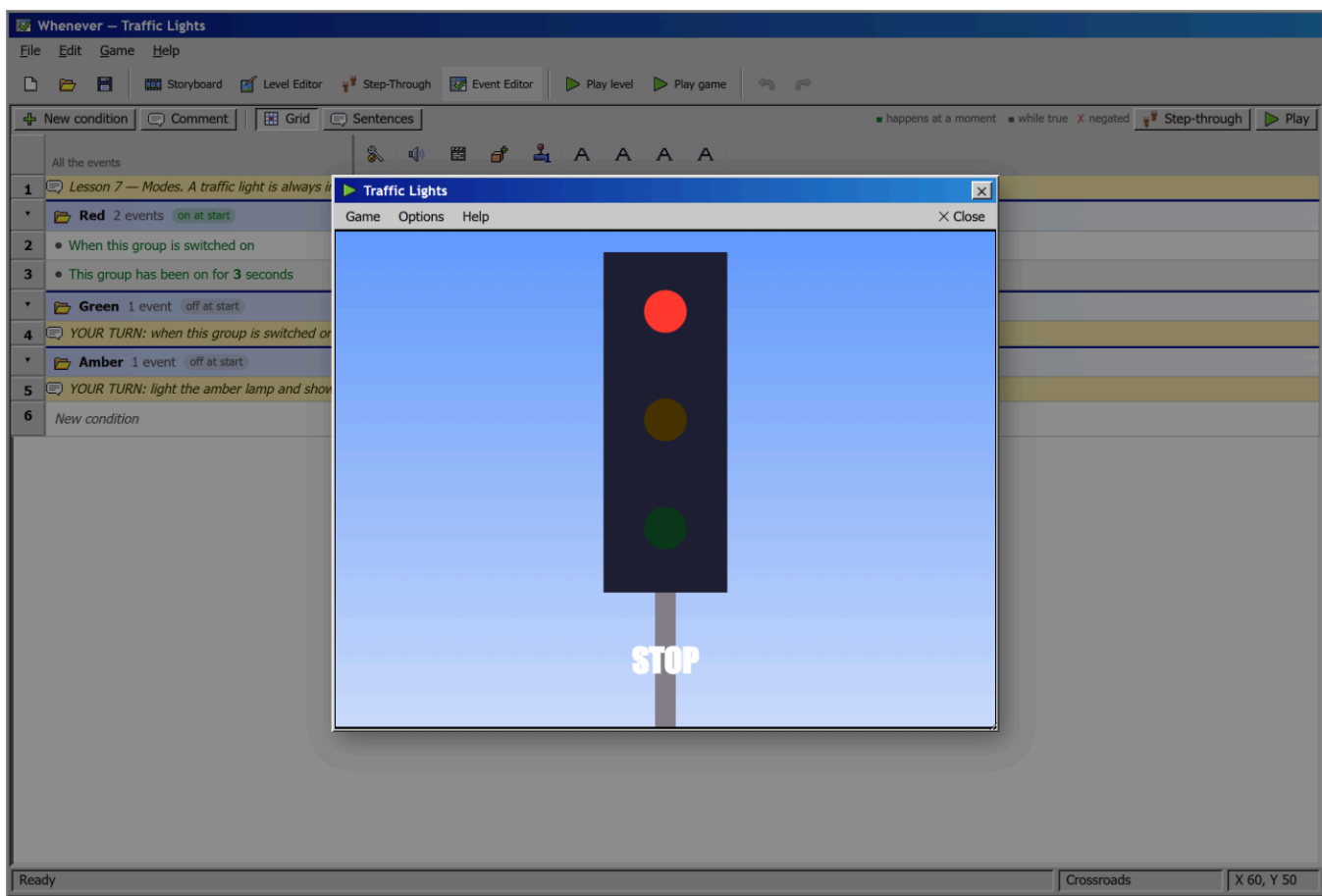


FIG 7.2 The red light, with STOP on the sign

Activity

25 MIN

- 1 **The Green mode.** Right-click the **Green** heading and choose **New event in this group**. Choose **Special** ▶ **Groups of events** ▶ **When this group is switched on**.
- 2 In that row, right-click under **Green Lamp** and choose **Set colour of text...**: a bright green. Dim the **Red Lamp** and **Amber Lamp** the same way. Under **Sign**, choose **Set text to a value...** and type "GO", with the quotes. (Shortcut: drag the ticks from the Red group's first event to copy them, then change the colours.)
- 3 Add a second event in the Green group: **Special** ▶ **Groups of events** ▶ **After this group has been on for...** 3 seconds. Under **Special: Switch a group off** (Green) and **Switch a group on** (Amber).
- 4 **The Amber mode.** Build it the same way: the amber lamp lit and *GET READY*, then after 1 second switch Amber off and Red on.
- 5 Play and watch the whole cycle. In the Step-Through Editor, the debugger's **Watch** list shows which group is on.

■ Stretch

- **A crossing button.** Add an object to be the button. In the **Green** group only: user clicks on the button → switch Green off and Amber on. Pressing it while the light is red does nothing, because that rule lives in the Green mode.
- **UK lights.** Add a *Red and amber* mode between Red and Green, with both lamps lit for 1 second.

■ Check for understanding

"When the light is amber, which groups are switched on?" (Only Amber.) "Why does the crossing button do nothing when the light is red?"

■ Common mix-ups

- ✗ **Forgetting to switch the old mode off.** Two modes are then on at once: the lamps fight and the timings go wrong.
- ✗ **Timing with Timer** ▶ **Every...** It counts from the start of the frame, not from when the mode began, so the colours get uneven times. Use **After this group has been on for...**
- ✗ **Missing quotes.** Text in **Set text to a value...** needs quotes: `"GO"`.

A group that is switched on starts running on the next tick. That is why one rule can't race through several modes in a single moment.

■ Answer key

	Group	Condition	Actions
1	Red	● When this group is switched on	✓ Red Lamp bright red ✓ Amber Lamp and Green Lamp dim ✓ Sign = "STOP"
2	Red	● This group has been on for 3 seconds	✓ Switch off Red ✓ switch on Green
3	Green	● When this group is switched on	✓ Green Lamp bright green ✓ Red Lamp and Amber Lamp dim ✓ Sign = "GO"
4	Green	● This group has been on for 3 seconds	✓ Switch off Green ✓ switch on Amber
5	Amber	● When this group is switched on	✓ Amber Lamp bright amber ✓ Red Lamp and Green Lamp dim ✓ Sign = "GET READY"
6	Amber	● This group has been on for 1 second	✓ Switch off Amber ✓ switch on Red



STARTER GAME

Whenever – Bug Hunt

File Edit Game Help

New Condition Comment Grid Sentences

- Storyboard Level Editor Step-Through Event Editor Play level Play game
- Step-through Play

All the events	[Icons]
Lesson 8 — Bug hunt. This game has FOUR bugs. Play it and write down what goes wrong. Then use the debugger (bottom of the Step-Through Editor) to find each one.	
Big Ball leaves the play area on the left	✓
Big Ball leaves the play area on the right	✓
Big Ball leaves the play area at the top	✓
Big Ball collides with Bat	✓
Big Ball collides with Red Brick	✓ ✓
Big Ball collides with Blue Brick	✓ ✓
Big Ball leaves the play area at the bottom + Big Ball is stopped	✓ ²
Score of Player 1 ≥ 200	✓
Score of Player 1 = 1000	✓
Number of lives of Player 1 reaches 0	✓
New condition	

Ready | Brick Wall | X 60, Y 50

FIG 8.1 Bug Hunt's events: four of them hide a bug

■ Objectives

- ✓ Describe a bug precisely: expected, actual, and when it happens.
- ✓ Use the debugger to find the event that causes a bug.
- ✓ Fix one thing at a time, and test after each fix.

Key words:

bug

debug

expected

actual

test

cause

symptom

■ Warm-up: spot the bug 5 MIN

Read out instructions for making toast with a bug in them: *Put the bread in the toaster. Spread the butter. Switch the toaster on.* Pupils say what will go wrong and which step to move.

■ Demonstration 10 MIN

Show the debugger at the bottom of the Step-Through Editor:

- **Speed** buttons slow the game to $\frac{1}{2}$, $\frac{1}{4}$ or $\frac{1}{10}$.
- **Pause** and **One tick** move a sixtieth of a second at a time.
- Each event in the list **glows** when its actions run.
- **Watch** shows the score, the lives and how many of each object there are.

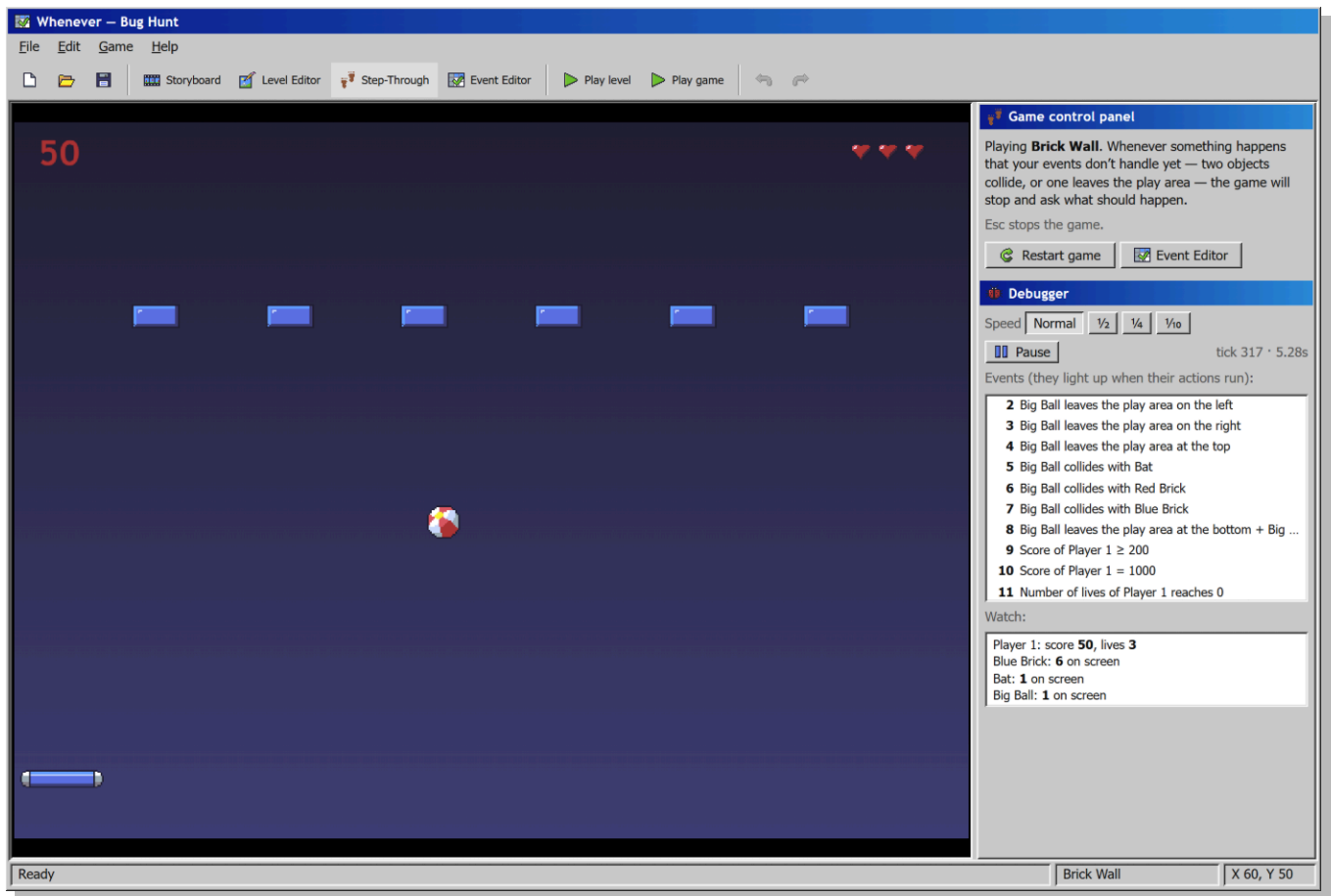


FIG 8.2 The debugger during Bug Hunt: the red bricks have all gone after one hit on a blue brick

Activity 25 MIN

Pairs keep a bug log:

What I expected	What happened	Which event?	My fix	Fixed?

- 1 Play for two minutes and fill in the first two columns for every problem you notice.
- 2 For each bug, play in the Step-Through Editor and watch the events. Which one glows at the moment it goes wrong? Which one should glow but doesn't?
- 3 Fix one bug, play to test, then move on to the next.

Stretch

Plant a bug for your partner. Save a copy first, change one event, and swap. Can your partner describe the symptom and find it?

■ Check for understanding

Each pupil writes one bug in the form: "I expected ... but ... happened, because event ..."

■ Common mix-ups

- ✗ **Fixing the symptom, not the cause.** For example, adding a new event to destroy blue bricks, rather than correcting event 7.
- ✗ **Changing several things at once.** Then nobody knows which change worked.
- ✗ **Thinking a bug is in the event that looks wrong.** Sometimes the cause is a condition that is never true, so the event never glows at all.

■ Answer key

Symptom	Event	Cause	Fix
Hitting a blue brick makes every red brick vanish, and blue bricks never break.	7	The action destroys Red Brick instead of Blue Brick. No condition picked a red brick, so all of them go.	Delete the tick under Red Brick and add Destroy under Blue Brick.
Missing the ball costs no life, and the ball doesn't come back.	8	The event also needs <i>Big Ball is stopped</i> , which is never true.	Delete that condition.
At 200 points the lives shoot up into the hundreds, and hearts fill the top of the screen.	9	It was meant to give one bonus life, but <i>Score ≥ 200</i> is a state, so it adds a life every tick.	Add Special ▶ Limit conditions ▶ Run this event once .
Knocking out every brick doesn't win.	10	It waits for a score of exactly 1000, but 12 bricks give 600.	Use <i>Have all Red Brick been destroyed?</i> + <i>Have all Blue Brick been destroyed?</i>

Lesson 9



Make it yours

Pupils plan a change to a finished game, build it in small steps, test it with other players, and share it as a web page. This lesson works well spread over two sessions.

STARTER GAME

Asteroid Dodge. A small finished game. Fly the spaceship with the arrow keys, dodge the rocks, and score a point for every second you survive. It uses ideas from earlier lessons: a family (*Rocks*), a limit condition (two seconds of safety after a hit), and objects created during play (the explosion).

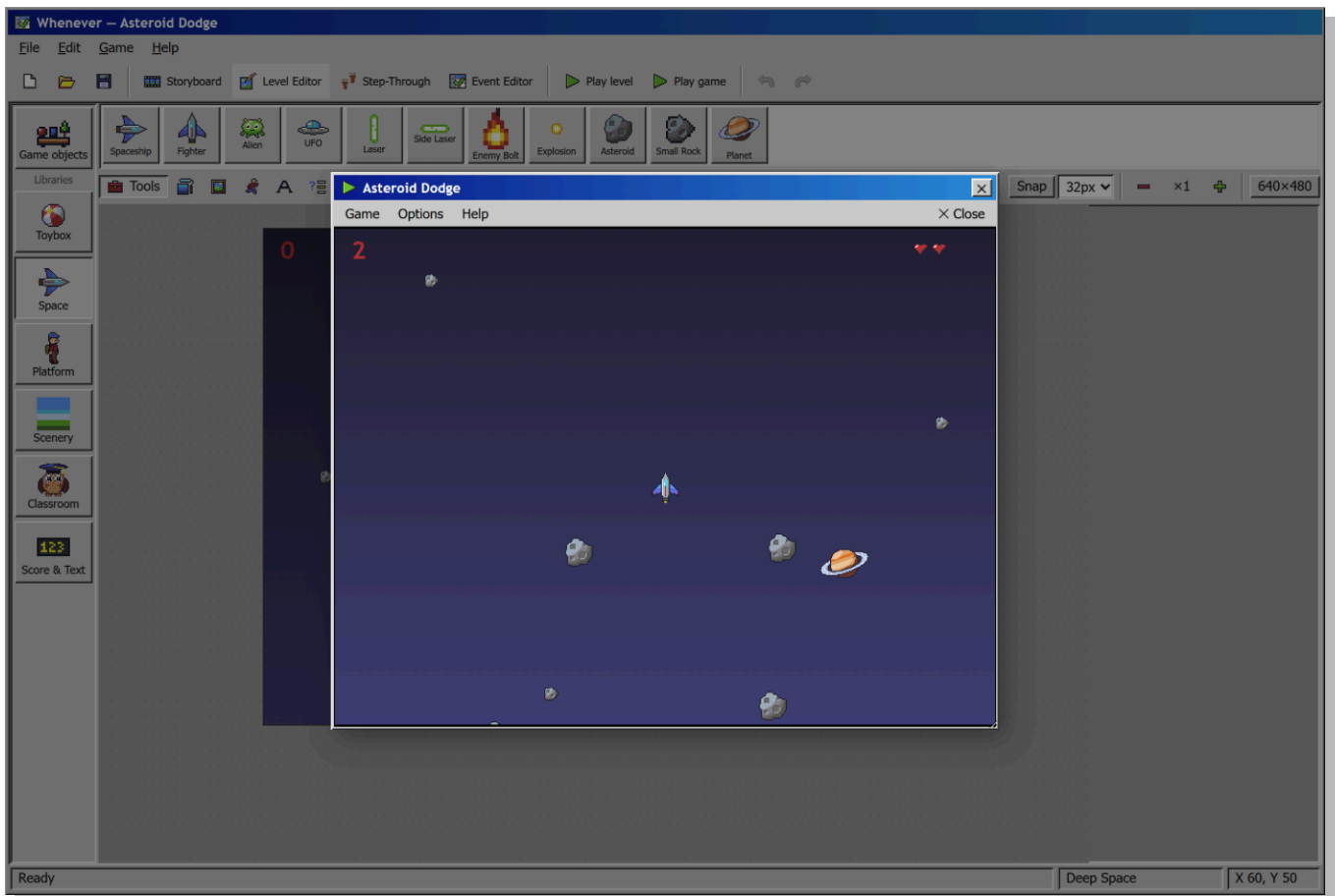


FIG 9.1 Asteroid Dodge being played: the spaceship, asteroids, small rocks and a planet

■ Objectives

- ✓ Plan a change to a game, writing new rules as sentences first.
- ✓ Build and test in small steps, one rule at a time.
- ✓ Give and use feedback, and share a finished game.

Key words:

design

remix

test

feedback

iterate

publish

■ Warm-up: play and pick apart 10 MIN

Pairs play Asteroid Dodge for three minutes, then open the Event Editor. Can they find the family, the two seconds of safety, and the rule that makes the explosion disappear? Which earlier lesson does each come from?

■ Plan 10 MIN

Each pair fills in a design brief on paper:

Question	Our answer
Who is the game for?	
How do you win? How do you lose?	
What is our twist?	
Three new rules, written as Whenever sentences	

■ Build 25 MIN, OR A WHOLE SESSION

Pick ideas from the table, or invent your own. Add one rule, play, and only then add the next.

Idea	How hard	How
New pictures for the ship or rocks	Easy	Double-click the object and draw in the Animation editor.
Stars to collect	Medium	Place a Star (Toybox); when the Spaceship collides with it, destroy it and add 10 points.
Shooting	Medium	Place a Laser (Space) in the holding area. Whenever Player 1 presses Fire 1, the Spaceship shoots the Laser. When a Laser collides with any Rocks, destroy both.
A second level	Medium	Duplicate the frame on the Storyboard and add more rocks. When the score reaches 30, go to the next frame.
A high score	Harder	A global value <i>Best</i> , saved on the computer (see lesson 5 and Coin Quest).
A title screen and game over	Harder	Groups of events for each mode (see lesson 7).

■ Test and share 10 MIN

- 1 Swap computers with another pair. Play their game and give *two stars and a wish*: two things you liked and one suggestion.
- 2 Make one change based on the feedback.
- 3 Choose **File** ► **Save stand-alone game...** to make a single web page. Share it on the school network or website.

■ Check for understanding

Each pair presents in one minute: "Our game is about ... One rule we are proud of is: Whenever ..."

■ Common mix-ups

- ✗ **Building everything before testing.** Ten new rules at once make bugs hard to find. Test after each rule.
- ✗ **Shots need an object to shoot.** The Laser must be in the game (the holding area is a good place) before **Shoot an object...** can use it.
- ✗ **Sharing safely.** Stand-alone games contain only what pupils typed into them. Use first names or class names in the Author box if the game will go online.

■ What's in the starter

	Conditions	Actions
1	● any Rocks leaves the play area	✓ Make any Rocks wrap around the play area
2	● Spaceship leaves the play area	✓ Make Spaceship wrap around the play area
3	● Every 1 second	✓ Add 1 to the score of Player 1
4	● Spaceship collides with any Rocks ● Restrict actions for 2 seconds	✓ Subtract 1 from the lives ✓ create an Explosion at the Spaceship ✓ play Bang
5	● Explosion's animation has finished	✓ Destroy Explosion
6	● Number of lives of Player 1 reaches 0	✓ End the game

Assessment

Use the exit questions and pupils' games to judge each idea on three levels. The *Secure* column is the goal for most pupils by the end of the course.

Idea (lesson)	Getting there	Secure	Going further
Events (1)	Adds an action in the Step-Through Editor with help.	Explains any event as <i>whenever ...</i> , <i>do ...</i> , and builds one alone.	Predicts what a new event will do before playing it.
Reading rules (2)	Reads an event aloud.	Predicts what a short set of rules will do.	Spots a rule that will cause a problem before running it.
And (3)	Adds two conditions, sometimes in separate events.	Puts conditions that must all be true into one event.	Uses Negate, or two events for <i>or</i> .
Moments and states (4)	Notices that green and black conditions look different.	Explains why a black condition repeats, and fixes it.	Uses a state on purpose, such as <i>Always</i> to keep a display up to date.
Variables (5)	Uses the score.	Chooses a counter, an alterable value or a global value for a job, and says why.	Combines variables in an expression, such as <code>"Times played: " + Global("Played")</code> .
Which one? (6)	Notices that some actions affect every copy.	Predicts which copies an action will affect.	Uses families and <i>Pick one at random</i> deliberately.
Modes (7)	Switches a group on or off.	Builds a mode with <i>When this group is switched on</i> and a timed switch to the next.	Designs a new mode from a state diagram.
Debugging (8)	Says that something is wrong.	Describes a bug as expected versus actual, and finds the event responsible.	Fixes causes rather than symptoms, one change at a time, and plants bugs for others.
Design (9)	Makes one change to a game.	Plans rules as sentences, then builds and tests them one at a time.	Acts on feedback and shares a finished game.

Handouts



Handouts

Copy these for pairs. Each one goes with a lesson.

Rule cards

Lesson 2: Reading rules aloud

Moment or state?

Lesson 4: Moments and states

Bug log

Lesson 8: Finding bugs

Design brief

Lesson 9: Make it yours



Rule cards

Name

Class

"What kind of game is this? Draw what the screen looks like."

A

RULE CARD

Every 1.5 seconds, make a new balloon just below the bottom of the screen, at a random place from left to right.

B

RULE CARD

Whenever someone clicks a balloon, destroy that balloon, add 10 to the score, and play a pop.

C

RULE CARD

Whenever a balloon floats off the top, destroy it and take away a life.

D

RULE CARD

Whenever the score is 200 or more, go to the next screen.

E

RULE CARD

Whenever the lives reach 0, end the game.

?

RULE CARD

Your own rule:

Whenever



Moment or state?

Name

Class

Cut out the cards. Sort them into **moment** and **state**.

collides with	is overlapping	upon pressing a key
score of Player 1 > 100	every 2 seconds	repeat while a key is pressed

Moment (green)

Happens at a moment. True for one instant.

State (black)

True while something lasts. Checked again every tick.



Bug log

Name

Class

Play for two minutes. Fill in the first two columns for every problem you notice.

What I expected	What happened	Which event?	My fix	Fixed?
				☐
				☐
				☐
				☐
				☐
				☐
				☐
				☐
				☐

I expected ... but ... happened, because event ...



Design brief

Name

Class

Who is the game for?

How do you win? How do you lose?

What is our twist?

Three new rules, written as **Whenever** sentences

1 **Whenever**

2 **Whenever**

3 **Whenever**

Two stars and a wish





Wish