

Game Builder: Change the Rules with Variables

Ages 9-14 · about 45 minutes, in one sitting

Your child plays a short platform game, changes one number, the jump power, and finds the player stuck in a valley it climbed out of a minute ago. They learn that gravity, jump and speed are variables the game reads, find the same numbers in the game's code, and tune the game to feel the way they want. At the end you play it and guess what they changed.

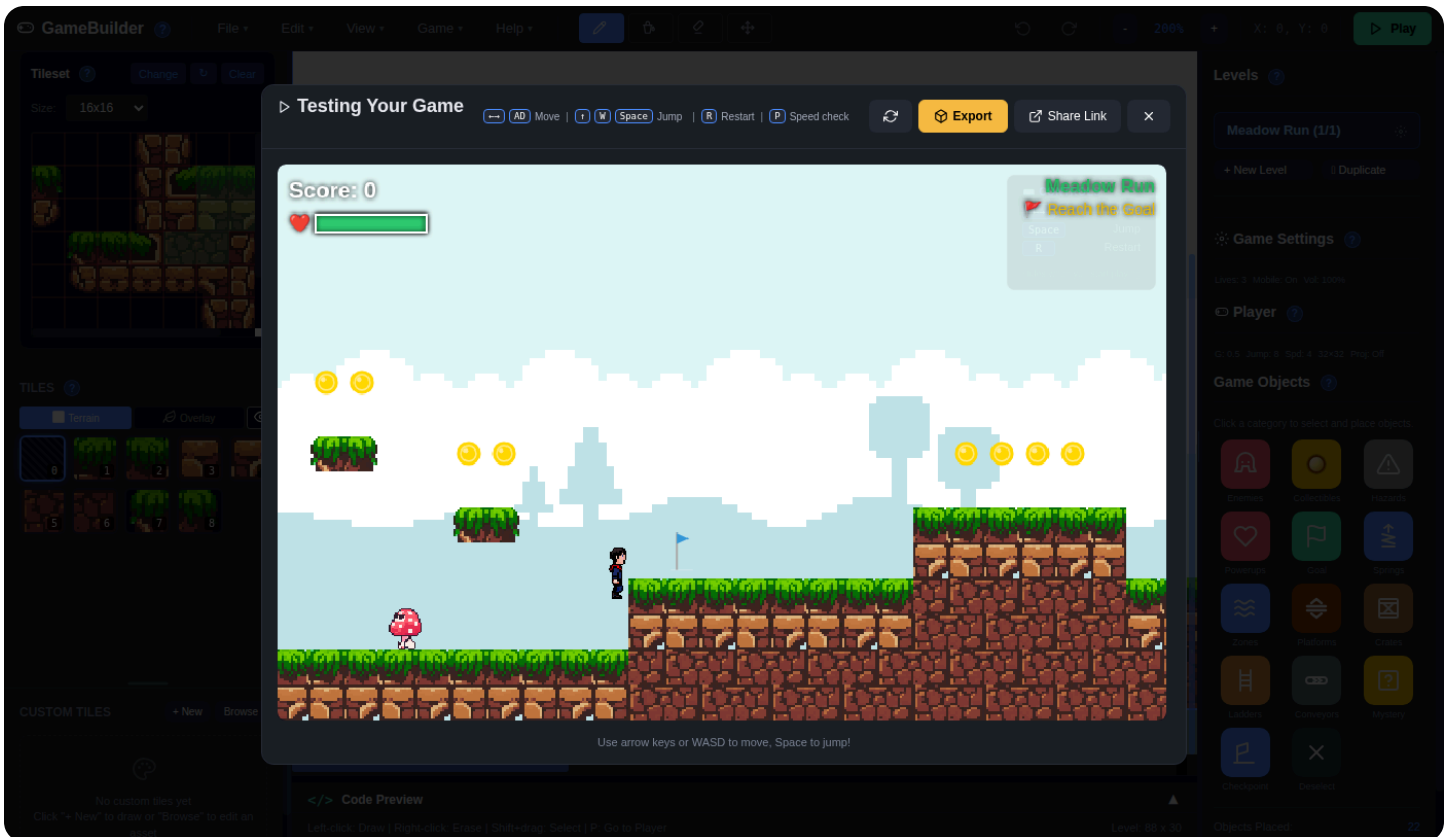
What you need

- A computer or Chromebook with a keyboard and a current browser. The arrow keys run and Space jumps.
- The lesson link: mytekdev.com/try/gamemaker?lesson=change-the-rules-with-variables&ref=fam-gamebuilder-1. No account, no email and nothing to install.
- Optional: the activity sheet at the end of this plan, printed, and a pencil, for writing each prediction down before testing it.

Before you start: try it once yourself

Break and fix Meadow Run yourself once before you start, about ten minutes. The arrow keys run, Space jumps, R starts the level over and Esc goes back to the editor.

1. **Open the lesson.** Go to mytekdev.com/try/gamemaker?lesson=change-the-rules-with-variables&ref=fam-gamebuilder-1. Meadow Run opens in the editor with a short guide along the bottom; press **Play**, run to the star, then press Esc.
2. **Read the variables.** Click **Player** on the right. The **Movement** tab shows **Gravity** 0.5, **Jump Power** 10 and **Move Speed** 4.
3. **Break it.** Set **Jump Power** to **8**, press **Done**, then **Play**, and run right. The player drops into the low valley with the mushroom and cannot jump back out of it.
4. **See it in the code.** Set **Jump Power** back to **10** and press **Done**. Click the **Code Preview** bar under the level, then click **Game Settings** on the right: the code shows `player.velocityY = -10` next to jump power and `+= 0.5` next to gravity; press **Done**.
5. **Gravity.** Set **Gravity** to **0.8** and play: the player is trapped in the valley again. Then try **0.3**, where the player floats and jumps much higher, and set it back to **0.5**.



Jump Power 8: the same level, and the player cannot get out of the valley.

If you see something else: Code Preview shows the gravity and jump code only after **Game Settings** is clicked, not **Player**. Esc switches between the editor and the game, so a second press starts the game again. The line under **Player** on the right shows the current G, Jump and Spd: check it after every test, because a number not set back spoils the next one.

The ideas, in plain words

A variable is a name a program uses for a value it keeps, so the program can read the value wherever it needs it. In Game Builder, Gravity, Jump Power and Move Speed are variables. Each rule of the game is written once and reads those names, so changing one number changes that rule everywhere in the game, on every level, at once. The level itself, the tiles and the objects, is separate: it does not change when a number does.

The rules run 60 times a second, once a frame. Every frame, gravity is added to the player's falling speed. When Space is pressed and the player is on the ground, its upward speed is set to the Jump Power, and while an arrow key is held, its sideways speed is set to the Move Speed. The rules and the level have to fit each other: the valley in Meadow Run has walls 2 tiles tall, a Jump Power of 10 climbs them, and 8 climbs less than 2 tiles. A bigger gravity does the same thing from the other side, by taking the speed away sooner.

YOUR CHILD MIGHT ASK	A GOOD ANSWER
Why is the jump power negative in the code?	On a screen, up means a smaller y. A negative speed moves the player up the screen.
Did I break the game?	No, the code still works. The numbers no longer fit the level, so you can fix the number or fix the level.
Why does bigger gravity make the jump lower?	Gravity pulls the player down every frame. Pulling harder stops the rise sooner, so the jump is lower.
Why not just put the number in each place it is used?	Then you would have to find and change every copy. With a variable you change it once, and everything that reads it changes too.

Do it together

1. Open mytekdev.com/try/gamemaker?lesson=change-the-rules-with-variables&ref=fam-gamebuilder-1 together. Meadow Run opens in the editor with a short guide along the bottom; follow it together. The arrow keys run, Space jumps, R starts the level over and Esc goes back to the editor. The guide has your child play to the star, open **Player**, set **Jump Power** to 8 and play again.

If you get stuck: Esc switches between the editor and the game, so a second press starts the game again.

You'll know it worked when: Running right, the player drops into the low valley with the mushroom and cannot jump back out.

2. Talk about what happened: the level did not change, only one number did. Gravity, Jump Power and Move Speed are variables, named numbers the game reads every time the player moves. The valley walls are 2 tiles tall; a Jump Power of 10 climbs them, and 8 climbs less than 2. Press Esc, click **Player**, set **Jump Power** back to 10 and press **Done**.

You'll know it worked when: The line under **Player** on the right shows Jump 10 again, and your child can say why 8 got the player stuck.

3. Find the numbers in the code. Click the **Code Preview** bar under the level, then click **Game Settings** on the right. Press **Done** when you have looked.

If you get stuck: Code Preview shows the gravity and jump code only after **Game Settings** is clicked, not **Player**.

You'll know it worked when: The code shows `player.velocityY = -10` next to jump power and `+= 0.5` next to gravity: the same numbers as in the Player box.

4. Predict, then test. Before touching the app, your child says or writes what Gravity 0.8 will do. Then click **Player**, set **Gravity** to 0.8, press **Done**, press **Play** and run to the valley. Do the same with 0.3, then set Gravity back to 0.5.

If you get stuck: Check the line under **Player** after every test, because a number that was not set back spoils the next test.

You'll know it worked when: Gravity 0.8 traps the player too, because it pulls harder and the jump is lower. At 0.3 the player floats and jumps much higher.

5. Now your child makes the game their own, with three rules: change at least two numbers, one at a time; press **Play** after every change; and a player must still be able to reach the star.

You'll know it worked when: The game feels different, and your child can still run it from the start to the star.

6. The real question: when a change makes some part impossible to get past, ask your child to name two different ways to fix it. One is to change the number back a little. The other is to change the level: pick a tile under **Tiles** on the left and click on the level to draw it, for example a step in the valley wall. Right-click erases.

You'll know it worked when: Your child names both, a number fix and a level fix. That means they see that the rules and the level work together, and that a variable is one place to change a rule for the whole game.

7. Trade seats. Your child presses **Play** first, because the Player box shows the numbers. You play their game for about 3 minutes without pressing Esc and guess which numbers they changed, and whether each went up or down. Then they tell you what really changed. Finish with the big question: how can changing one number change a whole game?

You'll know it worked when: Your child explains each change they made and what it did to the way the game feels.

Talk about it

Ask these while you work together. A child who has understood the lesson can answer them without help:

- Which number did you change, and what did you predict?
- Why does Jump Power 8 trap the player in the valley?
- Gravity 0.8 traps the player too. Why does a bigger gravity make the jump lower?
- Show me the jump power in Code Preview (click Game Settings, not Player).
- Your player cannot get past this part. Name two different ways to fix it.

The last one is the real test. A child who can fix it by changing the number or by changing the level understands that the rules and the level work together, and that a variable is one place to change a rule for the whole game.

For younger and older kids

Younger: For ages 7 and 8, do the gravity test side by side, and when they make the game their own, change only Jump Power.

Older: For teens, find all three variables in Code Preview, explain in writing why Gravity 0.8 and Jump Power 8 both trap the player, and write a short design note: the feel they wanted, and the numbers that made it.

Keep going

- Make a level part only your new jump can handle: if you raised Jump Power, draw a wall the old jump could not climb.
- Try Jump Mode, Double Jump, in the same Movement tab, and change the level to need it.
- Change Friction to 1 (ice). What happens when you let go of the arrow? Why does it not matter in the air?

Bring it into other subjects

- Science: gravity on the Moon is about one sixth of gravity on Earth. Set Gravity to 0.1, the smallest it goes, and play. Is a Moon level easier or harder? (Answers vary: the player jumps very high and floats, so landing on a small platform takes care.)
- Math: measure how high each Jump Power reaches, in tiles. Does doubling the jump double the height?
- English: write the rules of your game for a player who has never played, in three sentences or fewer, including how high and how fast the player moves.
- Computer science: open Code Preview and click Game Settings. Find the line for move speed and say what happens to it when the left arrow is pressed. (The code sets the speed to minus the Move Speed, so the player moves left.)

For your records

Ready to copy into an hour log, a portfolio or a transcript.

LESSON	Game Builder: Change the Rules with Variables
SUBJECTS	Computer science, Game design
TIME	45 minutes
WHAT WE DID	Changed the variables that set a platform game's gravity, jump power and speed, predicting each result before testing it, and found the same numbers in the game's code. Tuned the game to a feel of their own and fixed the parts that broke, by changing a number or changing the level.
SKILLS PRACTICED	variables in a program; predicting, then testing; fixing a problem two ways; tuning how a game feels

Activity sheet (optional)

Name: _____ Date: _____



Open the app: scan the QR code, or go to mytekdev.com/try/gamemaker?lesson=change-the-rules-with-variables

Essential question: How can changing one number change a whole game?

- 1. **Play it.** Press **Play**. Run with the arrow keys, jump with Space, and reach the star. **Esc** goes back.
- 2. **Name the variables.** Click **Player**. Write the three numbers on the Movement tab: Gravity ____ Jump Power ____ Move Speed ____
- 3. **Predict, then test.** Fill in each row of the table: write your prediction first, then change the number, press **Done** and **Play**.
- 4. **Make it yours.** Change at least two numbers. A player must still reach the star. Write your final numbers below.

CHANGE	I PREDICT...	WHAT HAPPENED
Gravity 0.8		
Gravity 0.3		
My change 1:		
My change 2:		

My final numbers: Gravity ____ Jump Power ____ Move Speed ____

- 1. Which prediction surprised you most, or was wrong? What did you learn from it?

- 2. My playtester guessed I changed: _____ (higher or lower?)

Quick check

Name: _____

1. With Jump Power 8 the player is stuck in the valley. What changed? (A) the level, (B) the value of a variable the game reads, (C) the player's picture.
2. Gravity goes from 0.5 to 0.8. The jump gets: (A) higher, (B) lower, (C) the same.
3. In someone's game, the player cannot climb a wall. Name two different ways to fix it.

Answers

QUESTION	ANSWER	WHY
1	B	The level is the same. Only the number the game reads for the jump changed, and the new jump is too low for the valley wall.
2	B	Gravity pulls the player down. A bigger gravity pulls harder, so the player stops rising sooner.
3	Any two of: raise Jump Power; lower Gravity; make the wall shorter; add a step or a platform in front of it	Any two fixes that would each work.

If your state or umbrella school asks, this lesson covers CSTA 2017 1B-DA-07, CSTA 2017 1B-AP-15, CSTA 2017 2-AP-17.

Game Builder is a mytekDev app, open source under the GNU General Public License 3.0. The goal star in Meadow Run is from Kenney's UI Pack, released under CC0.

Terms of use

This lesson was made by mytekDev. You may print and copy it for the children you teach at home, and change it for them. You may not sell it, share it with other families, or post it publicly.

The app this lesson uses is free and runs in the browser, with nothing to install and no account needed. It is at mytekdev.com.

© 2026 mytekDev. All rights reserved.