

Creating a Platformer in Game Maker Part 2

Foundations of Interactive Game Design
Prof. Jim Whitehead
February 9, 2007



Creative Commons
Attribution 2.5

creativecommons.org/licenses/by/2.5/

UC SANTA CRUZ



Upcoming Events and Assignments

- **Game Concept Document**

- ▶ Due today
- ▶ May still turn in later, but will suffer a late penalty

- **Gamelog**

- ▶ Due today by midnight
- ▶ Game of your choice

- **Work Breakdown and Schedule**

- ▶ Due next week, on Wednesday
- ▶ Looking for a list of tasks that need to be completed
- ▶ Estimate for how long each task will take
- ▶ Date by which the task should be completed
- ▶ Who will perform the task
- ▶ Slack time to recover if things go wrong
- ▶ Be conservative! Your game will take longer than you think.

RPG Maker/Game Maker/Art help

- **Game Maker reduced cost license keys**
 - ▶ See me after class to pay for keys
 - ▶ I will sell all unclaimed keys starting today
 - ▶ Full version of Game Maker on ITS machines as well
 - ▶ Game Maker temporarily not selling licenses for version 6.1
 - ▶ Version 7 out soon apparently

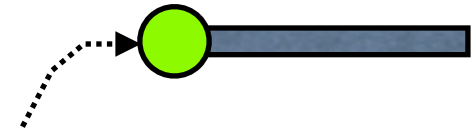
Key Mechanics of Platformers

- To create a platform game, need to
 - ▶ Handle collision with platforms
 - ▶ Handle jumping
 - ▶ Scrolling view in large game world
 - ▶ Interactions with other enemies and items in gameworld
- Today, will focus on
 - ▶ improved collision detection
 - ❖ Fixing problems identified in the last lecture
 - ▶ moving platforms

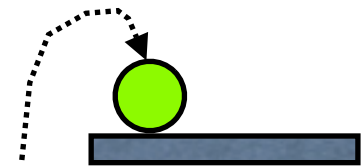
Summary of Collision Problems

- Need to handle collisions correctly

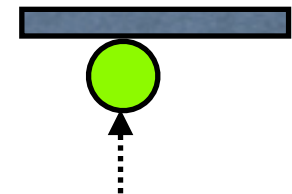
- ▶ Not going into the side of platforms



- ▶ Correctly landing on the top of platforms



- ▶ Correctly handling jumps up into platforms

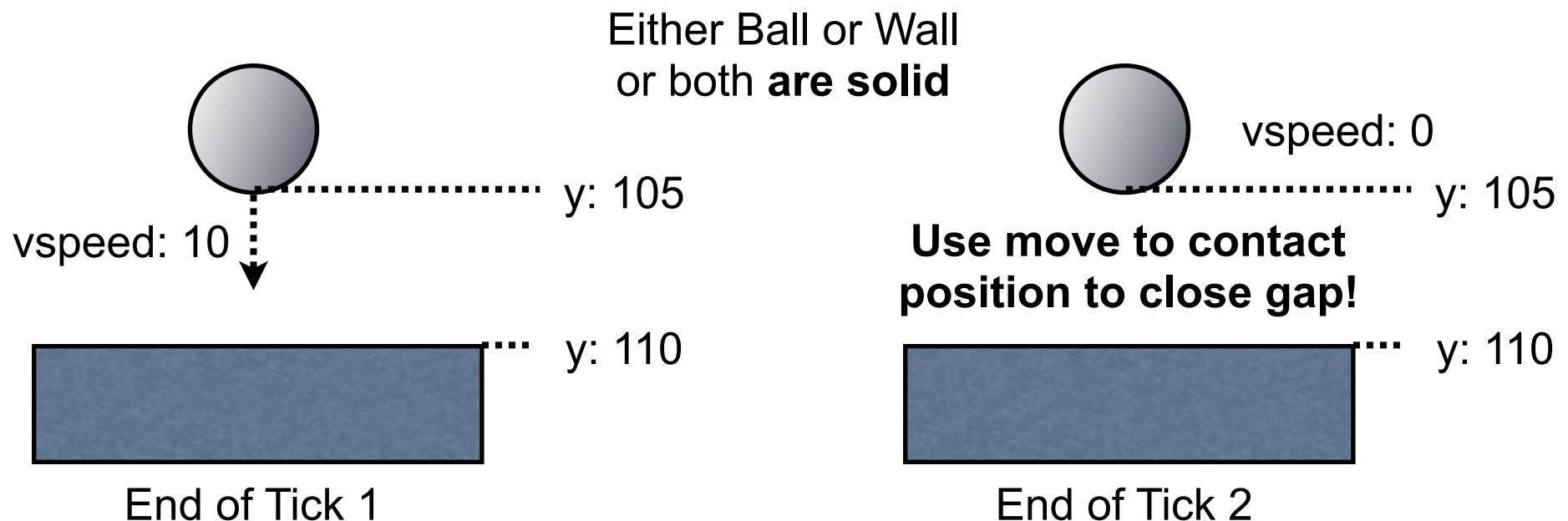


- ▶ Game Maker does not easily give you a way to determine which situation is occurring.

- ❖ Simple approach leads to sticking to side of platform, and infinite jump on underside of a platform

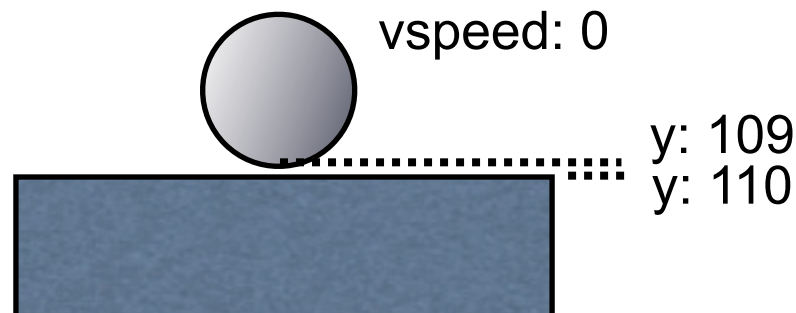
Collision Landing Problem

- Problem: sprite does not go all the way down to the platform on a collision
 - ▶ Happens when ball or wall are solid
 - ▶ Due to solidity, object reset to previous position before collision event generated
 - ▶ Collision event does not update the position.
 - ▶ Need to move ball that last little bit to have smooth transition
 - ▶ Use “move to contact position” (in middle of Move tab on Object)



Move to Contact Position

- Moves an object instance in a particular direction until there is contact with an object
 - ▶ The object is placed right before a collision would occur
- If the object begins in a collided state, no motion occurs
- Can specify the direction, and maximum amount of movement
 - ▶ Maximum amount of movement is useful to avoid unexpected side-effects



End of Tick 2 if using move to contact position

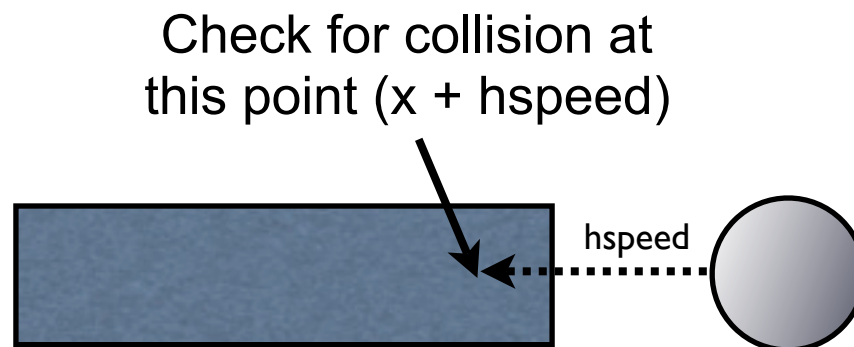
Avoiding Sticking to Side of Platform

- Problem recap:
 - ▶ Collision of ball with wall sets vspeed to 0
 - ▶ But, every tick the ball moves into the wall horizontally
 - ❖ hspeed isn't affected by collision detection, and needs to be
 - ▶ However, don't want to just arbitrarily set hspeed to 0
 - ❖ Would cause object to come to complete stop when landing on a platform
 - ❖ Motion feels less fluid
- Want to check whether there is a collision in the horizontal direction
- Use “If position is collision free” conditional
 - ▶ On Control tab of Object
- Allows checking of whether there is a collision at some position
 - ▶ Often want to use a position relative to the current object

Avoiding Sticking to Side of Platform (2)

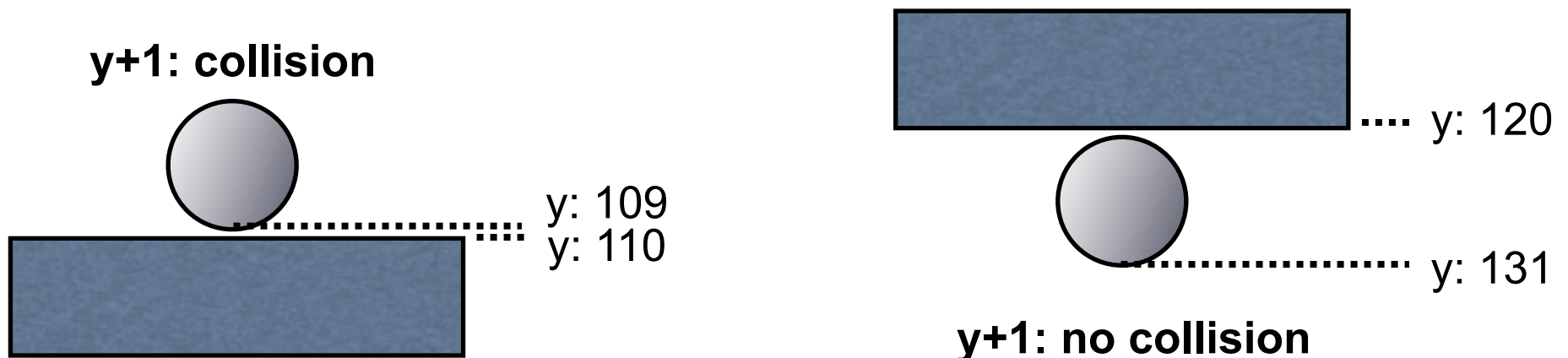
- Approach

- ▶ Check to see if there is a collision in the location the ball is about to move to horizontally
 - ❖ Ball or Wall must be solid
 - ❖ Use “If a position is collision free”
 - ❖ Applies to: self
 - ❖ $x: \text{hspeed}$
 - adds current horizontal velocity to x , essentially computing next location
 - ❖ $y: 0$
 - ❖ objects: only solid
- ▶ If so, set hspeed to 0
 - ❖ Or bounce



Avoiding Double Jump Under Platform

- Problem recap: collisions with the underside of a platform act just the same as collisions to the top side
 - ▶ In particular, collision causes the jumping state machine to reset to the not-jumping state
 - ▶ This allows the player to jump again, even though they are in mid-air
- Need a better approach for determining when to reset state machine back to non-jumping
 - ▶ One idea is to check for a collision with location just under the ball
 - ▶ In figure below, if we check for a collision at $y+1$ ($109+1 = 110$), there is a collision

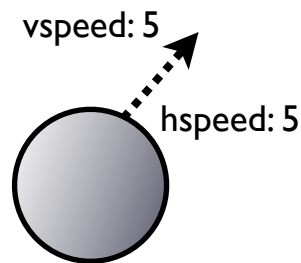


Replication Problem

- Now the collision detection is finally working well
 - ▶ For one pair of objects
- What happens if there is more than one kind of wall
 - ▶ Ugh, need to repeat all of the collision detection code
 - ▶ Would be nice if there was some standard bit of code that would handle all platform collision detection

All-in-one Platformer Collision Handler

- Jump to the position held by the object at start of step
- Check if the position $(x+h\text{speed}, y+v\text{speed})$ is a collision
 - ▶ That is, at the position the object wants to move to this step, will it run into anything?
 - ▶ If not, then just move it to the new position
 - ❖ $x = x+h\text{speed}, y = y+v\text{speed}$



Any collision at new location? No.
Move to new location.

All-in-one Platformer Collision Handler (2)

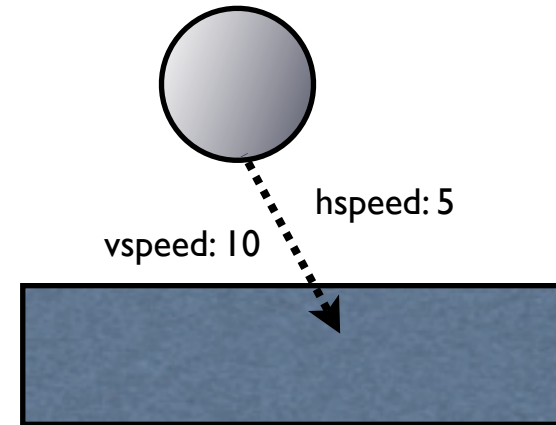
- Else, if there is a collision at new position
 - ▶ Check first for horizontal collisions
 - ❖ If the object has horizontal velocity (if `hspeed` does not equal 0)
 - ▶ If so, may or may not have a collision in x
 - ▶ Want to keep moving until there is a collision or move entire `hspeed` if none
 - ❖ Use “Move to contact position”
 - Direction is: $180 * (\text{hspeed} < 0)$
 - The “`hspeed < 0`” part is 1 if moving left (`hspeed` will be negative, hence less than zero), and 0 if moving right (`hspeed` is positive, hence greater than zero)
 - Maximum movement: `abs(hspeed)`
 - The absolute value of the `hspeed`. Take absolute value, since the direction is now indicated by an angle, and not the sign
 - Against: solid objects
 - ▶ After moving, still don't know if there was a collision.
 - ❖ May have moved the maximum movement without collision

All-in-one Platformer Collision Handler (3)

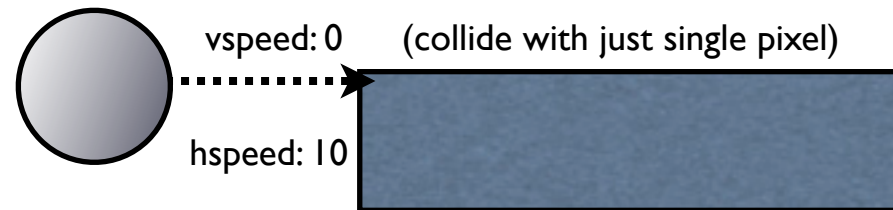
Three cases:

Assumes x check and
x movement comes first

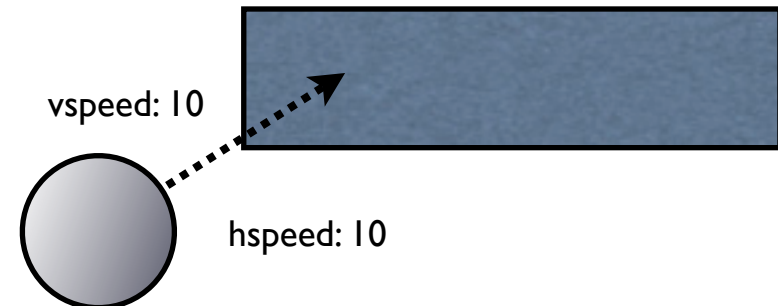
Collision in y, but not x



Collision in x, but not y



Collision in x and y



All-in-one Platformer Collision Handler (4)

- Need to check for horizontal collision
 - ▶ Use the “check one pixel over” approach
 - ▶ For platforms, could be a collision on either side
 - ❖ Want to check the side on the direction the object is moving
 - ❖ Use “If a position is collision free”
 - Applies to: self
 - x: $\text{sign}(\text{hspeed})$: returns -1 for $\text{hspeed} < 0$ (moving left), or 1 for $\text{hspeed} > 0$ (moving right)
 - y: 0 (are only checking for collisions in x)
 - objects: Only solid
 - Relative: yes
 - NOT: yes (want to check to see if there is a collision, the negative of collision free)
 - ❖ If there is a horizontal collision, set the hspeed to 0
 - Will stop object right before the object being collided with (a wall)

All-in-one Platformer Collision Handler (5)

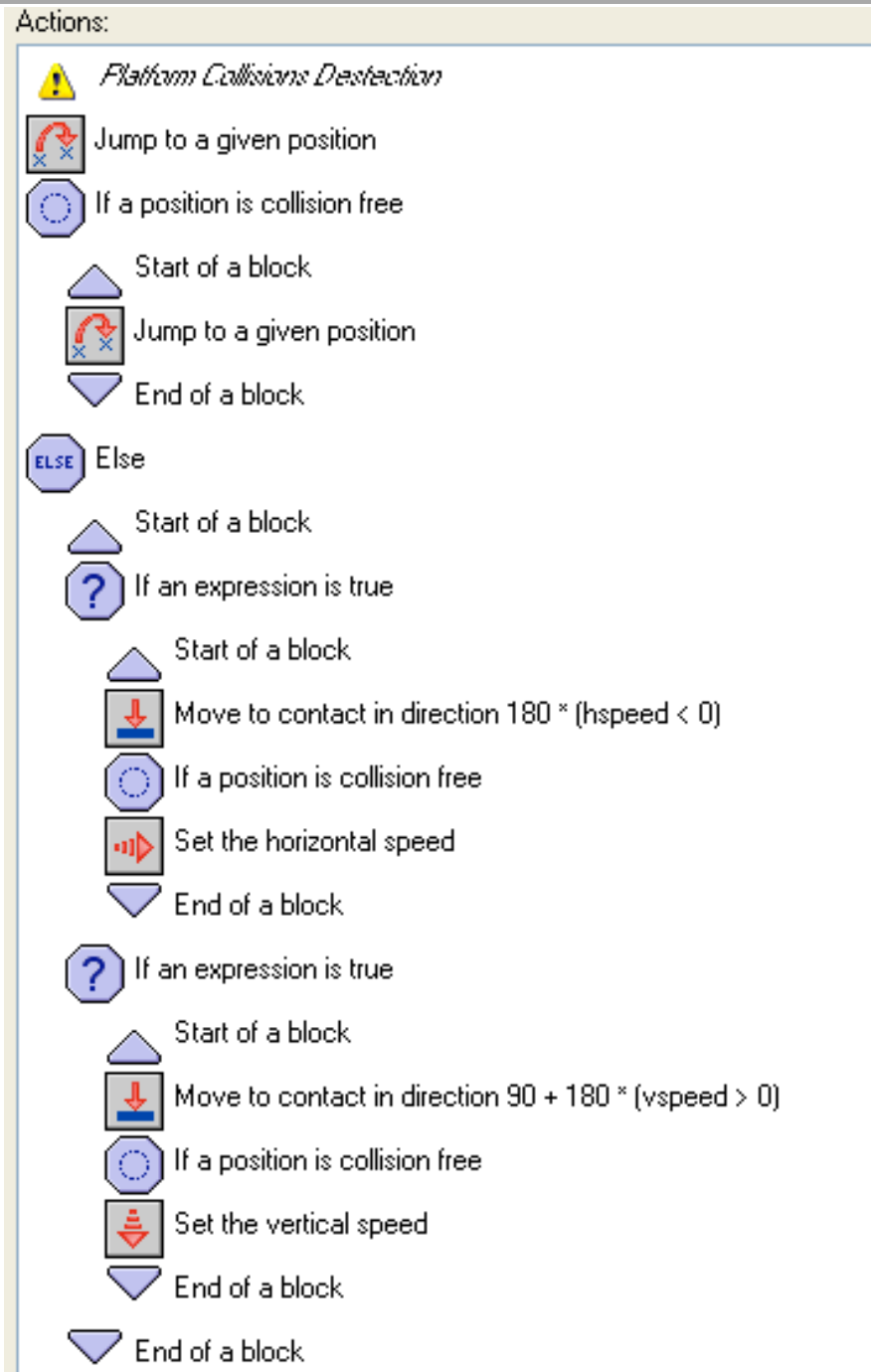
- Now do similar checks in the vertical direction
 - ▶ Move to contact position vertically
 - ❖ Direction: $90 + 180 * (\text{vspeed} > 0)$
 - $\text{vspeed} > 0$ is 1 if vspeed is positive (moving downward), resulting in angle of 270
 - ❖ Maximum movement: $\text{abs}(\text{vspeed})$
 - Convert signed speed into unsigned speed, since direction is now an angle, not a sign
 - ❖ Against: solid objects
 - ▶ May not have had a collision, so check for collision one pixel over
 - ▶ If so, set vertical speed to 0

All-in-one Platformer Collision Handler (6)

These actions go into the End Step event for the player object.

All walls in the game must be solid.

No longer need collision event for any walls.



Making a Moving Platform

- Two main issues in creating a moving platform
 - ▶ Need to make platform move along a consistent path
 - ▶ Once the player has landed on the platform, the players speed must match that of the platform
- Approach
 - ▶ Use a **path** to control movement of the platform
 - ▶ Add a check for a collision with the player object to the platform
 - ❖ Update player movement if true

Paths

- An advanced feature
 - ▶ Have I mentioned you really want the advanced version?
- A predefined pathway enemies can follow
 - ▶ Pattern of movement of enemies in Shmup
 - ▶ Pattern of movement of enemies through a level in a Platformer
 - ❖ Think about barrels in Donkey Kong: always follow one of small set of paths
- To create
 - ▶ Add.. Add Path
 - ▶ Define set of points on the path
 - ▶ Can have different speeds at different points on path
- To make work
 - ▶ Connect with an object
 - ❖ In create event, use action to start path (on Move tab)
 - ▶ Can specify **repeating**/one-time, speed, **relative** or absolute coords

Paths for Platforms

- When making a back and forth path for a platform
 - ▶ Recommend making a closed path
 - ❖ That is, go out some distance, then come back to where you started
 - ▶ In Create event for moving platform object
 - ❖ Use “Set path for the instance”
 - ❖ At end: continue from start
 - “reverse” appears to be buggy
 - ❖ Relative: relative
 - ▶ If you use “reverse” I have seen platforms start oscillating wildly after about 5-6 back-and-forth movements

Updating Player Position

- We're using the all-in-one collision detection approach in the player object
 - ▶ Would like to avoid adding additional complexity to this
 - ▶ So, add collision detection logic to moving platform
 - ▶ A bit backwards: have platform determine if player has landed on it
 - ❖ Usually think of the player colliding with the platform

Updating Player Position (2)

- Add to “End Step” event
 - ▶ Needs to be End Step to come later than the player object’s position updates
 - ▶ Use “If there is an object at a position”
 - ❖ Blue ball in Octagon on Control tab of MovingPlatform object
 - ❖ Object: player object (Ball)
 - ❖ x: 0
 - ❖ y: -1
 - ❖ Relative: yes
 - ❖ NOT: no
 - ▶ Checks to see if the player is in any of the pixels immediately above the platform

Updating Player Position

- If yes
 - ▶ Use “Set the value of a variable” (grey square with “Var” on Control)
 - ▶ Object: player object (Ball)
 - ▶ Variable: x (this is the Ball’s x value we’re updating)
 - ▶ Value: $\text{MovingPlatform.x} - \text{MovingPlatform.xprevious}$
 - ❖ Add this step’s movement for the moving platform to the x value of the player
 - ❖ Causes the player to move with the platform
 - ❖ Note that path following does not change the speed of the platform object
 - So, cannot just add MovingPlatform.hs peed, even though that may seem reasonable.
 - ▶ Relative: yes
- This approach still has problems with side-collision stickies
 - ▶ Can use approaches discussed previously in this lecture to address

Game Maker Platform Resources

- Tutorial on making Platformers with Game Maker
 - ▶ <http://www.gamemaker.nl/tutorials/platform.zip>
- Improved platformer engine for use with Game Maker
 - ▶ <http://www.pages.drexel.edu/~mfp27/platformengine/>
 - ▶ No experience with using this, looks promising