Finishing the Platformer

Jump and shoot!

Game Programming Foundations

Last modified 16/03/15 by Sam Cartwright

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Finishing the Platformer

Jump and shoot!

Topics

- What’s missing?
- Adding Enemies
- Adding Bullets
- Climbing Ladders
- Adding a Score / Lives
- Adding Triggers

What do we have so far?

- Run and jump
- Platforms / Ladders
- Animations
- Music
- Side-scrolling
- But it’s not a game… yet

What's Missing?

- Watch this amazing YouTube for a post-mortem on a classic platformer
- Language warning

https://www.youtube.com/watch?v=8FpigqfcvlM&list=PLu5a9-aw8CA_xRgSUtUjcEem6d_cJk9Mx

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

What can we add?

- Enemies
- Bullets
- Climbing (this one’s a little tough)
- Triggers
- Score and Life counter
- (Your game may or may not need any or all of these)

Enemies

- enemy.js is very similar to player.js
- Enemies move automatically
  - First move right, if can move no further move left
- Same collision detection to detect end of platform
- If an enemy is not placed on a platform, it won't move

Enemy - Update

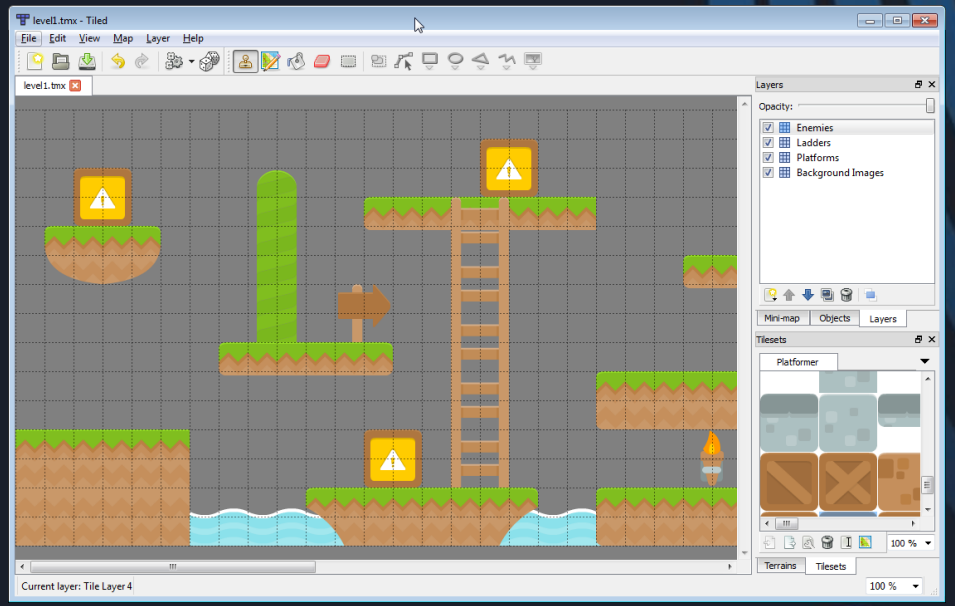
```
pause timer = 0
move right = true

def update (deltaTime)
    update sprite

    if pause timer greater than 0
        // pause enemy before changing direction
        decrease pause timer by delta time
    else
        acceleration x = 0

        tile X = pixelToTile( enemy position X )
        tile Y = pixelToTile( enemy position Y )

        overlap X = enemy position X % tile width
        overlap Y = enemy position Y % tile height

        // find which cells contain platforms
        cell = is platform at coordinates (tile X, tile Y)
        cell right = is platform at coords (tile X, tile Y)
        cell down = is platform at coords (tile X, tile Y)
        cell diagonal = is platform at coords (tile X, tile Y)

        if move right is true
            if cell diagonal not empty and cell right is empty
                    // path clear, keep moving right
                    increment acceleration X
                else
                    // path blocked, pause then turn back
                    stop moving
                    set move right to false
                    set pause timer
            end if

            if move right is false
                if cell down is not empty and cell is empty
                    // keep moving left
                    decrease acceleration x
                else
                    // path blocked, pause then turn back
                    stop moving
                    start enemy moving right
                    set pause timer
            end if

            update velocity by adding acceleration, clamp to range
            update enemy position x by adding velocity
    end
```

Enemies

- We need to manually create and place enemies in the world
- Can we add enemies in our level editor?

Enemies

- Create a new layer
- Pick any unused tile
- Place tiles where enemies should appear
- Modify main.js to create an enemy wherever this tile appears



aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Enemies – Initialization

```
for each row in the enemies layer
    for each column in the enemies layer
        if the enemies layer at tile (column, row) is not empty
            create a new enemy
            set the enemy position to tile (column, row)
            add the enemy to the enemies list
        end if
    end for
end for
```

- This code would go in the initialize() function in main.js
- To update / draw, in main.js step through the enemies array using a for loop

Enemies – Player Collision

- Kill the player if colliding with enemy
- Modify the run() function in main.js
- We covered testing the intersection of 2 rectangles with Asteroids (Basic Linear Math & Collision Detection)

```
for each enemy in enemies array
    update the enemy
    if player is alive
         if player's collision rectangle intersects the enemy's rect
             set player.isAlive to false
         end if
    end if
end for
```

Bullets

- Create a bullets array

- Bullet logic is easy:
  - Keep going left/right until an enemy or screen edge is hit
  - Bullet update function should just move the bullet left/right
  - Use the run() function in main.js to kill bullets when offscreen

- We should also update the player to face the direction of movement
  - Drawing a flipped image in JS is expensive
  - Need a new sprite map with reversed images

Bullets

- When the bullet is created, set its velocity to move left/right
- The bullet's update() is then very simple

```
def update (deltaTime)
    update sprite
    position X = position X + deltaTime * velocity X
end
```

- Use main.js run() to kill bullets when offscreen

```
hit = false
for each bullet in the bullets array
    update the bullet

    // check if the bullet went offscreem
    // remember we are also scrolling the world based on the player's
    // pos (so we need to find the bullet's screen coords)
    if bullet position X - worldOffsetX is less than 0 or
        the bullet position X - worldOffsetX greater than screen width
        hit = true
    end if

    // also check if the bullet hit an enemy
    if bullet's collision rectangle intersects the enemy's rect
        remove the enemy from the enemies array     // use splice()
        hit = true
    end if

    if hit is true
        remove the bullet from the bullets array    // use splice()
    end if
end for
```

Shooting

- Update the player's sprite to face the direction of movement
  - Use a sprite sheet with containing the flipped frames
  - Create the animations
  - When the left/right key pressed, change the animation
- Keep track of which direction the player is facing
  - When shooting, set the bullet velocity based on current direction

Shooting

```
def update (deltaTime)
    if not alive
        return

    update sprite

    left = false
    right = false
    jump = false

    if left key is down
        left = true
        set current direction to LEFT
        set current animation to walking left if not already
    else if right key is down
        right = true
        set current direction to RIGHT
        set current animation to walking right if not already
    else
        if not jumping and not falling
            if direction is LEFT
                set animation to idle left if not already
            else
                set animation to idle right if not already
            end
        end
    end

    if up key is pressed
        jump = true
    end

    decrease bullet timer by delta time
    if space key is down and bullet timer is less than 0
        create a new bullet
        set bullet velocity based on current direction
        add bullet to bullets array
        set bullet timer to 0.5 seconds
    end

    // remainder of update function deals with running
    // and jumping on the platforms and does not change
    ...
end
```

Climbing

- Only the player.js file will need updating
- Add a state machine to the player
- If near a ladder and pressing up/down, switch to climb state
- If in climb state and reach the end of a ladder, switch to run&jump state

Climbing

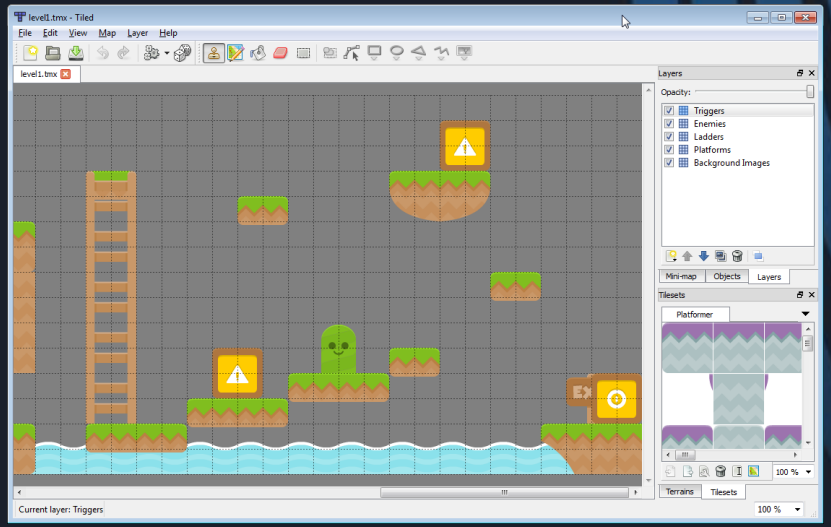
```
def updateRunJumpState

    // mostly stays the same, but we add some new logic
    // at the end of the function

    if right is false and left is false and falling is false
        // player is not moving or falling, but could be
        // jumping (because we use the up key for both
        // jumping and climbing)

        cell = cell at tile coordinates (tile X, tile Y) on
               ladder layer
        cell right = cell at tile coords (tile X + 1, tile Y) on
               ladder layer
        cell down = cell at tile coords (tile X, tile Y + 1) on
               ladder layer
        cell diag = cell at tile coords (tile X + 1, tile Y + 1)
               on ladder layer

        // check if we are standing at the bottom of a ladder
        if cell is not empty or cell right is not empty
            if the up key is being pressed
                set the state to the climb state
                set the animation to the climb animation
                return
            end
        end

        // check if we are standing at the top of a ladder
        if cell down is not empty or cell diag is not empty
            if the down key is being pressed
                set the state to the climb state
                set the animation to the climb animation
                return
            end
        end
    end
end
```

Climbing

```
def updateClimbState
    climb up = false
    climb down = false

    if up key is down
        climb up = true
        update sprite  // only update sprite when moving
    end
    if down key is down
        climb down = true
        update sprite
    end

    if velocity Y is greater than 0
        was moving up = true
    end
    if velocity Y is less than 0
        was moving down = true
    end

    reset acceleration Y
    if climb up is true
        acceleration Y = acceleration Y - ACCEL
    else if was moving up
        reset velocity Y
    end

    if climb down is true
        acceleration Y = acceleration Y + ACCEL
    else if was moving down
        reset velocity Y
    end

    add acceleration Y to velocity Y and clamp to range
    add velocity to position

    calculate tile X,Y using player's position
    cell = cell at tile coord (tile X, tile Y) for ladder layer
    cell right = cell at tile coord (tile X + 1, tile Y) for
                 ladder layer
    cell down = cell at tile coord (tile X, tile Y + 1) for ladder
                 layer
    cell diag = cell at tile coord (tile X + 1, tile Y + 1) for
                 ladder layer

    if velocity Y is greater than 0 or was moving down
        if cell down and cell are empty or cell diag and
         cell right are empty
            switch to run jump state and return
        end
    else if velocity Y is less than 0 or was moving up
        if cell and cell down are empty or cell right and
         cell diag are empty
            switch to run jump state and return
        end
    end
end
```

Triggers

- Add a new layer to the level
- Add a tile for the ‘game over’ trigger
- Create a collision map for this layer
- When the player collides with this trigger, switch to a ‘game over’ state



aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Triggers

- Modify initialize() in main.js to build a collision map for the trigger layer
- This check could go in the player's update function
- Add a state machine to the game to allow for a 'game over' state

```
var tile X = pixel to tile (player position x)
var tile Y = pixel to tile (player position y)

if cell at tile coord (tile X, tile Y) on the trigger layer is not 0
    switch game state to game over state
end
```

Score

- Keep a score variable (in main.js)
- When a bullet hits an enemy, increment the score
- Draw the score somewhere on th

```
// draw the score
context.font = "24px Verdana";
context.fillStyle = "#FF0";
context.fillText("SCORE: " + score, 20, 40);
```

Summary

- We did a lot this lesson
  - Adding enemies, bullets, triggers, score, and climbing ladders
- You should be able to see how we build a game by breaking it down and solving individual problems one at a time
- Your game might not need everything here
  - A lot can be achieved through clever level design
  - could you design a platformer that doesn't need enemies / bullets / climbing / triggers / anything else?

Questions?

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

References

- JavaScript Tutorial. 2016. JavaScript Tutorial. [ONLINE] Available at: http://www.w3schools.com/js/default.asp. [Accessed 01 March 2016].