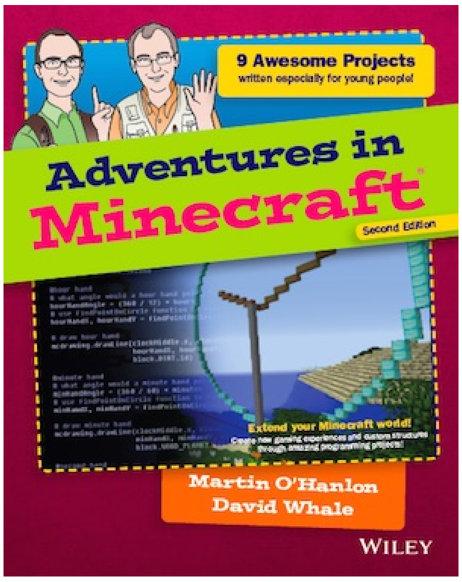
Adventures in Minecraft (2nd Edition)

Program Listings

Martin O'Hanlon and David Whale

November 2017



MARTIN O'HANLON has been designing and programming computer systems for all of his adult life. His passion for programming and helping others to learn led him to create the blog <Stuff about="code" />(www.stuffaboutcode.com) where he shares his experiences, skills and ideas. Martin regularly delivers presentations and workshops on programming Minecraft to coders, teachers and young people with the aim of inspiring them to try something new and making programming fun.

DAVID WHALE writes computer programs for devices you wouldn't imagine have computers inside them. He was bitten by the computer programming bug aged 11 when he was at school, and still thoroughly enjoys writing software and helping others to learn programming. He runs a software consultancy business in Essex, but also regularly volunteers for The Institution of Engineering and Technology (The IET) helping in schools, running weekend computing clubs, judging schools competitions, and running programming workshops for young people at community events all around the UK. You can follow his adventures on his twitter page at http://www.twitter.com/whaleygeek.

Programs from: *Adventures in Minecraft (2nd Edition)* by Martin O'Hanlon and David Whale, Wiley, 2017

http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1119439582.html

https://github.com/AdventuresInMinecraft/code-files

Contents

1 Adventure 1: Hello Minecraft World

1.1 Hello Minecraft World

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()
mc.postToChat("Hello Minecraft World")
```

2 Adventure 2: Tracking Your Players as They Move

2.1 Where Am I

```
# This program works out where you are in the Minecraft world.
# It prints a message on the Minecraft chat, with your coordinates.

# The Minecraft API has to be imported before it can be used
import mcpi.minecraft as minecraft

# To communicate with a running Minecraft game,
# you need a connection to that game.
mc = minecraft.Minecraft.create()

# Ask the Minecraft game for the position of your player
pos = mc.player.getTilePos()

# Display the coordinates of the players position
mc.postToChat("x="+str(pos.x) + " y="+str(pos.y) +" z="+str(pos.z))

# END
```

2.2 Where Am I 2

```
# This program works out where you are in the Minecraft world.
# It uses a game loop to repeatedly display your position
# on the Minecraft chat.

# The Minecraft API has to be imported before it can be used
import mcpi.minecraft as minecraft

# The time module allows you to insert time delays
import time

# To communicate with a running Minecraft game,
# you need a connection to that game.
mc = minecraft.Minecraft.create()

# A game loop will make sure your game runs forever
while True:
    # delay, to prevent the Minecraft chat filling up too quickly
    time.sleep(1)

    # Ask the Minecraft game for the position of your player
    pos = mc.player.getTilePos()

    # Display the coordinates of the players position
    mc.postToChat("x="+str(pos.x) + " y="+str(pos.y) +" z="+str(pos.z))

# END
```

2.3 Welcome Home

```
# This program creates a magic doormat that says "welcome home"
# when you stand on it.

# Import necessary modules
import mcpi.minecraft as minecraft
import time

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# The main game loop
while True:
    # Short delay to prevent chat filling up too quickly
    time.sleep(1)

    # Get player position
    pos = mc.player.getTilePos()

    # Check whether your player is standing on the mat
    if pos.x == 10 and pos.z == 12:
        mc.postToChat("welcome home")

# END
```

2.4 Rent

```
# This program is a game using geo-fencing.
# You are challenged to collect objects from a field in the shortest
# time possible. All the time you are in the field you are charged
# rent. You have to collect/destroy blocks by spending the minimum
# amount of rent. If you stay in the field for too long, your player
# is catapulted up into the sky and out of the field, which makes the
# game harder to play and much more fun!

# Import necessary modules
import mcpi.minecraft as minecraft
import time

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# Define constants for the 4 coordinates of the geo-fence
X1 = 10
Z1 = 10
X2 = 20
Z2 = 20

# Constants for the place to move through when catapulted!
HOME_X = X2 + 2
HOME_Y = 10 # up in the sky
HOME_Z = Z2 + 2

# A variable to keep a tally of how much rent you have been charged
rent = 0

# A variable to hold the number of seconds the player is in the field
inField = 0

# The main game loop ticks round once every second
while True:
    # Slow the program down a bit, this also helps with timing things
    time.sleep(1) # the loop runs once every second

    # Get the players position
    pos = mc.player.getTilePos()

    # If the player is in the field, charge him rent
    if pos.x>X1 and pos.x<X2 and pos.z>Z1 and pos.z<Z2:
        # Charge 1 pound rent every time round the loop (1sec per loop)
        rent = rent + 1
        # Tell player how much rent they owe
        mc.postToChat("You owe rent:" + str(rent))
        # Count number of seconds player is in the field (1sec per loop)
        inField = inField + 1
    else:
        # Not inside the field...
        inField = 0 # ...so reset the counter to zero

    # If player in field for more that 3 seconds...
    if inField>3:
        mc.postToChat("Too slow!")
        # ...catapult player outside of the field
        mc.player.setPos(HOME_X, pos.y+HOME_Y, HOME_Z)

# END
```

3 Adventure 3: Building Anything Automatically

3.1 Block

```
# This program builds a block next to your player.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# Get the player position
pos = mc.player.getTilePos()

# Create a block in front of the player
mc.setBlock(pos.x+3, pos.y, pos.z, block.STONE.id)

# END
```

3.2 Build

```
# This program builds a block at an absolute coordinate

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# Build a block
mc.setBlock(1, 2, 3, block.DIAMOND_BLOCK.id)

# END
```

3.3 Dice

```
# This program builds the face of a dice near your player.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# Get the player position
pos = mc.player.getTilePos()

# Create 6 blocks in front of the player, that look like a dice
mc.setBlock(pos.x+3, pos.y,   pos.z,   block.STONE.id)
mc.setBlock(pos.x+3, pos.y+2, pos.z,   block.STONE.id)
mc.setBlock(pos.x+3, pos.y+4, pos.z,   block.STONE.id)
mc.setBlock(pos.x+3, pos.y,   pos.z+4, block.STONE.id)
mc.setBlock(pos.x+3, pos.y+2, pos.z+4, block.STONE.id)
mc.setBlock(pos.x+3, pos.y+4, pos.z+4, block.STONE.id)

# END
```

3.4 Tower

```
# This program builds a tall tower inside Minecraft, using a for loop.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# Get the player position
pos = mc.player.getTilePos()

# Build a tower 50 blocks high
for a in range(50):
    # Each block is at height: y+a
    mc.setBlock(pos.x+3, pos.y+a, pos.z, block.STONE.id)

# END
```

3.5 Clear Space

```
# This program clears some space near your player.
# This is so that it is then easier for you to build other things.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# Get the player position
pos = mc.player.getTilePos()

# Ask the user how big a space to clear
size = int(input("size of area to clear? "))

# Clear a space size by size*size*size, by setting it to AIR
mc.setBlocks(pos.x, pos.y, pos.z, pos.x+size, pos.y+size, pos.z+size,
             block.AIR.id)

# END
```

3.6 Build House

```
# This program builds a single house, with a doorway, windows,
# a roof, and a carpet.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block

# Connect to Minecraft
mc = minecraft.Minecraft.create()

# A constant, that sets the size of your house
SIZE = 20

# Get the players position
pos = mc.player.getTilePos()

# Decide where to start building the house, slightly away from player
x = pos.x + 2
y = pos.y
z = pos.z

# Calculate the midpoints of the front face of the house
midx = x+SIZE/2
midy = y+SIZE/2

# Build the outer shell of the house
mc.setBlocks(x, y, z, x+SIZE, y+SIZE, z+SIZE, block.COBBLESTONE.id)

# Carve the insides out with AIR
mc.setBlocks(x+1, y, z+1, x+SIZE-1, y+SIZE-1, z+SIZE-1, block.AIR.id)

# Carve out a space for the doorway
mc.setBlocks(midx-1, y, z, midx+1, y+3, z, block.AIR.id)

# Carve out the left hand window
mc.setBlocks(x+3, y+SIZE-3, z, midx-3, midy+3, z, block.GLASS.id)

# Carve out the right hand window
mc.setBlocks(midx+3, y+SIZE-3, z, x+SIZE-3, midy+3, z, block.GLASS.id)

# Add a wooden roof
mc.setBlocks(x, y+SIZE, z, x+SIZE, y+SIZE, z+SIZE, block.WOOD.id)

# Add a woolen carpet, the colour is 14, which is red.
mc.setBlocks(x+1, y-1, z+1, x+SIZE-2, y-1, z+SIZE-2, block.WOOL.id, 14)

# END
```

3.7 Build House 2

```
# This program builds a house with a doorway, windows, a roof
# and a carpet. It defines a house() function that you can easily
# reuse in other programs.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block

# Connect to Minecraft
mc = minecraft.Minecraft.create()

# A constant, that sets the size of your house
SIZE = 20

# define a new function, that builds a house
def house():
    # Calculate the midpoints of the front face of the house
    midx = x+SIZE/2
    midy = y+SIZE/2

    # Build the outer shell of the house
    mc.setBlocks(x, y, z, x+SIZE, y+SIZE, z+SIZE, block.COBBLESTONE.id)

    # Carve the insides out with AIR
    mc.setBlocks(x+1, y, z+1, x+SIZE-1, y+SIZE-1, z+SIZE-1, block.AIR.id)

    # Carve out a space for the doorway
    mc.setBlocks(midx-1, y, z, midx+1, y+3, z, block.AIR.id)

    # Carve out the left hand window
    mc.setBlocks(x+3, y+SIZE-3, z, midx-3, midy+3, z, block.GLASS.id)

    # Carve out the right hand window
    mc.setBlocks(midx+3, y+SIZE-3, z, x+SIZE-3, midy+3, z, block.GLASS.id)

    # Add a wooden roof
    mc.setBlocks(x, y+SIZE, z, x+SIZE, y+SIZE, z+SIZE, block.WOOD.id)

    # Add a woolen carpet, the colour is 14, which is red.
    mc.setBlocks(x+1, y-1, z+1, x+SIZE-2, y-1, z+SIZE-2, block.WOOL.id, 14)

# Get the players position
pos = mc.player.getPos()

# Decide where to start building the house, slightly away from player
x = pos.x + 2
y = pos.y
z = pos.z

# run the house() function that was defined by def house():
# this will build a house at x,y,z
house()

# END
```

3.8 Build Street

```
# This program builds a street of identical houses.
# It uses a for loop.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block

# Connect to Minecraft
mc = minecraft.Minecraft.create()

# A constant, that sets the size of your house
SIZE = 20

# define a new function, that builds a house
def house():
    # Calculate the midpoints of the front face of the house
    midx = x+SIZE/2
    midy = y+SIZE/2

    # Build the outer shell of the house
    mc.setBlocks(x, y, z, x+SIZE, y+SIZE, z+SIZE, block.COBBLESTONE.id)

    # Carve the insides out with AIR
    mc.setBlocks(x+1, y, z+1, x+SIZE-1, y+SIZE-1, z+SIZE-1, block.AIR.id)

    # Carve out a space for the doorway
    mc.setBlocks(midx-1, y, z, midx+1, y+3, z, block.AIR.id)

    # Carve out the left hand window
    mc.setBlocks(x+3, y+SIZE-3, z, midx-3, midy+3, z, block.GLASS.id)

    # Carve out the right hand window
    mc.setBlocks(midx+3, y+SIZE-3, z, x+SIZE-3, midy+3, z, block.GLASS.id)

    # Add a wooden roof
    mc.setBlocks(x, y+SIZE, z, x+SIZE, y+SIZE, z+SIZE, block.WOOD.id)

    # Add a woolen carpet, the colour is 14, which is red.
    mc.setBlocks(x+1, y-1, z+1, x+SIZE-2, y-1, z+SIZE-2, block.WOOL.id, 14)

# Get the players position
pos = mc.player.getTilePos()

# Decide where to start building the house, slightly away from player
x = pos.x + 2
y = pos.y
z = pos.z

# build 5 houses, for a whole street of houses
for h in range(5):
    # build one house
    house()
    # move x by the size of the house just built
    x = x + SIZE

# END
```

3.9 Build Street 2

```
# This program builds a street of houses with random carpets.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block

# A module to generate random numbers
import random

# Connect to Minecraft
mc = minecraft.Minecraft.create()

# A constant, that sets the size of your house
SIZE = 20

# define a new function, that builds a house
def house():
    # Calculate the midpoints of the front face of the house
    midx = x+SIZE/2
    midy = y+SIZE/2

    # Build the outer shell of the house
    mc.setBlocks(x, y, z, x+SIZE, y+SIZE, z+SIZE, block.COBBLESTONE.id)

    # Carve the insides out with AIR
    mc.setBlocks(x+1, y, z+1, x+SIZE-1, y+SIZE-1, z+SIZE-1, block.AIR.id)

    # Carve out a space for the doorway
    mc.setBlocks(midx-1, y, z, midx+1, y+3, z, block.AIR.id)

    # Carve out the left hand window
    mc.setBlocks(x+3, y+SIZE-3, z, midx-3, midy+3, z, block.GLASS.id)

    # Carve out the right hand window
    mc.setBlocks(midx+3, y+SIZE-3, z, x+SIZE-3, midy+3, z, block.GLASS.id)

    # Add a wooden roof
    mc.setBlocks(x, y+SIZE, z, x+SIZE, y+SIZE, z+SIZE, block.WOOD.id)

    # Add a woolen carpet, the colour is 14, which is red.
    mc.setBlocks(x+1, y-1, z+1, x+SIZE-1, y-1, z+SIZE-1, block.WOOL.id, 14)

    # Add a random carpet
    c = random.randint(0, 15)
    mc.setBlocks(x+1, y+1, z+1, x+SIZE-1, y+1, z+SIZE-1, block.WOOL.id, c)


# Get the players position
pos = mc.player.getTilePos()

# Decide where to start building the house, slightly away from player
x = pos.x + 2
y = pos.y
z = pos.z

# build 5 houses, for a whole street of houses
for h in range(5):
    # build one house
    house()
    # move x by the size of the house just built
    x = x + SIZE
```

```
# END
```

4 Adventure 4: Interacting with Blocks

4.1 Safe Feet

```
# This program works out if your feet are safe or not.
# If you are in the air or in water, your feet are not safe.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block
import time

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# Define a function to decide if your feet are safe or not
# This function will be reusable in other programs
def safeFeet():
  # Get the players position
  pos = mc.player.getTilePos()

  # Get the block directly below your player
  b = mc.getBlock(pos.x, pos.y-1, pos.z) # note: pos.y-1 is important!

  # Is the player safe?
  if b == block.AIR.id or b == block.WATER_STATIONARY.id or \
    b == block.WATER_FLOWING.id:
    mc.postToChat("not safe")
  else:
    mc.postToChat("safe")

# Game loop
while True:
  # Run the game loop once every half a second
  time.sleep(0.5)
  # Check if your feet are safe
  safeFeet()

# END
```

4.2 Magic Bridge

```
# This program builds a bridge under your feet as you walk around.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# Define a function to build a bridge if your feet are not safe
# This function will be reusable in other programs
def buildBridge():
  # Get the players position
  pos = mc.player.getTilePos()

  # Get the block directly below your player
  b = mc.getBlock(pos.x, pos.y-1, pos.z)
  # Is the player unsafe?
  if b == block.AIR.id or b == block.WATER_FLOWING.id or \
    b==block.WATER_STATIONARY.id:
    # If unsafe, build a glass block directly under the player
    mc.setBlock(pos.x, pos.y-1, pos.z, block.GLASS.id)

# Game loop
while True:
  # There is no delay here, this loop needs to run really fast
  # otherwise your bridge might not build quick enough and you will fall off
  buildBridge()

# END
```

4.3 Vanishing Bridge

```
# This program builds a bridge as you walk in the air or on water,
# and when your feet are safe again, the bridge magically disappears.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block
import time

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# Create a bridge list that is empty.
# There are no blocks in your bridge at the moment,
# so it always starts empty
bridge = []

# Define a function to build a bridge if your feet are not safe
# This function will be reusable in other programs
def buildBridge():
  # Get the position of your player
  pos = mc.player.getTilePos()

  # Get the block directly below your player
  b = mc.getBlock(pos.x, pos.y-1, pos.z)

  # Is the player safe?
  if b == block.AIR.id or b == block.WATER_FLOWING.id or \
    b == block.WATER_STATIONARY.id:
    # Player is unsafe, build a glass block directly under the player
    mc.setBlock(pos.x, pos.y-1, pos.z, block.GLASS.id)
    # Use a second list to remember the three parts of this coordinate
    # but note pos.y-1 as you want to remember the block position below
    coordinate = [pos.x, pos.y-1, pos.z]

    # Add the new coordinate to the end of your bridge list
    bridge.append(coordinate)

  elif b != block.GLASS.id:
    # Player is safe and not on the bridge
    if len(bridge) > 0:
      # There is still some bridge remaining
      # Remove the last coordinate from the bridge list
      coordinate = bridge.pop()

      # Set the block at that coordinate to AIR, to delete the bridge
      mc.setBlock(coordinate[0], coordinate[1], coordinate[2], block.AIR.id)

      # Delay a short time so that the bridge vanishes slowly
      time.sleep(0.25)

# Game loop
while True:
  # Don't run the game loop too fast, you might crash your computer
  # But don't run the game loop too slow, you might fall off the bridge
  time.sleep(0.25)

  # Decide what to do with the bridge
  buildBridge()

# END
```

4.4 Block Hit

```
# This program senses that a block has been hit.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block
import time

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# Work out the position of the player
diamond_pos = mc.player.getTilePos()

# Move the diamond slightly away from the player position
diamond_pos.x = diamond_pos.x + 1

# Build a diamond treasure block
mc.setBlock(diamond_pos.x, diamond_pos.y, diamond_pos.z,
            block.DIAMOND_BLOCK.id)

# Define a function that checks if the diamond treasure has been hit
# You can reuse this function in other programs
def checkHit():
  # Get a list of hit events that have happened
  events = mc.events.pollBlockHits()

  # Process each event in turn
  for e in events:
    # Get the coordinate that the hit happened at
    pos = e.pos

    # If the diamond was hit
    if pos.x == diamond_pos.x and pos.y == diamond_pos.y and \
      pos.z == diamond_pos.z:
      mc.postToChat("HIT")

# Game loop
while True:
  # Run the game loop once every second
  # Don't run it too fast or your computer might crash
  time.sleep(1)

  # Check to see if the diamond treasure has been hit
  checkHit()

# END
```

4.5 Sky Hunt

```
# This program is a treasure hunt game.
# It places treasure randomly near your player in the sky.
# You have to find this treasure in the fewest moves possible,
# because as you move you leave a trail of gold behind you.
# Each block of gold costs you points from your score.
# When you find the treasure, the gold trail melts away leaving
# holes in the ground for you to fall down, and another block
# of treasure is placed at a random location.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block
import time
import random # used generate random numbers

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# Create a variable to keep track of your score
score = 0

# Create a constant that sets how far away to create the treasure
RANGE = 5

# There is no treasure yet, but create the global variables to use later
# Set these to None as they have no value yet, but the variables need
# to be created here to prevent an error later
treasure_x = None
treasure_y = None
treasure_z = None


# Define a function that places a new piece of treasure
# It will place the treasure at a random location
# but this location will be close to your player
def placeTreasure():
  # This function will change the values in these global variables
  # so you must list them as global here
  global treasure_x, treasure_y, treasure_z

  # Get the position of the player
  pos = mc.player.getTilePos()

  # Calculate a random position for the treasure
  # This is within 'RANGE' blocks from the player position
  # and also always slightly higher up in the sky
  # Note how the treasure_ variables are given a new value here
  # which means they no longer have 'None' stored in them.
  # This signals to the main loop that the treasure is now placed
  treasure_x = random.randint(pos.x,   pos.x+RANGE)
  treasure_y = random.randint(pos.y+2, pos.y+RANGE)
  treasure_z = random.randint(pos.z,   pos.z+RANGE)

  # Place the treasure in the world at the new position
  mc.setBlock(treasure_x, treasure_y, treasure_z, block.DIAMOND_BLOCK.id)


# Define a function that checks if the diamond treasure has been hit
# You wrote this function in an earlier program, it is similar
def checkHit():
  # This function will change the values in these global variables
  # so you must list them as gloal here
  global score
  global treasure_x

  # Get a list of hit events that have happened
  events = mc.events.pollBlockHits()

  # Process each event in turn
  for e in events:
    # Get the coordinate that the hit happened at
    pos = e.pos

    # If the diamond treasure was hit
    if pos.x == treasure_x and pos.y == treasure_y and pos.z == treasure_z:
      mc.postToChat("HIT!")
      # Increase the score
      score = score + 10

      # Delete the treasure so it disappears
      mc.setBlock(treasure_x, treasure_y, treasure_z, block.AIR.id)

      # Set treasure_x to None so that the game loop knows that there
      # is no longer any treasure, and it will place a new block of
      # treasure when it is ready
      treasure_x = None


# Create a timer variable that counts 10 loops of the game loop
TIMEOUT = 10
timer = TIMEOUT

# Define a function that displays the homing beacon on the Minecraft chat
def homingBeacon():
  # This function changes the value in the timer variable
  # so it must be listed as global here
  global timer

  # Treasure will be present in the sky if the treasure_x variable
  # has a value. If it has no value (None), then there is no treasure.
  # Check to see if a piece of treasure has been placed
  if treasure_x != None:
    # There is a piece of treasure in the world somewhere

    # You only want to update the homing beacon every 10 times
    # round the game loop, so count down from 10
    timer = timer - 1
    if timer == 0:
      # Once you have counted 10 game loops, 1 second has passed
      # re-set the timer so that it starts counting another 10 game loops
      timer = TIMEOUT

      # Get the position of the player
      pos = mc.player.getTilePos()

      # Calculate a rough number that estimates distance to the treasure
      diffx = abs(pos.x - treasure_x)
      diffy = abs(pos.y - treasure_y)
      diffz = abs(pos.z - treasure_z)
      diff = diffx + diffy + diffz

      # Display score and estimate to treasure location on the Minecraft chat
      mc.postToChat("score:" + str(score) + " treasure:" + str(diff))
```

```
# Just like in your vanishingBridge.py program, create an empty bridge list
# there are no blocks in the bridge yet
bridge = []

# Define a function to build a bridge of gold blocks
# This is similar to the one you wrote in vanishingBridge.py
def buildBridge():
  # This function will change the value in the score variable
  # so it has to be listed as global here
  global score

  # Get the position of the player
  pos = mc.player.getTilePos()

  # Get the id of the block directly below the player
  b = mc.getBlock(pos.x, pos.y-1, pos.z)

  # Has the treasure been found and deleted?
  if treasure_x == None:
    # Are there still blocks remaining in the gold bridge?
    if len(bridge) > 0:
      # Remove the last coordinate of the bridge from the bridge list
      coordinate = bridge.pop()

      # Delete that block of the bridge by setting it to AIR
      mc.setBlock(coordinate[0], coordinate[1], coordinate[2], block.AIR.id)

      # Display a helpful countdown message on the Minecraft chat
      mc.postToChat("bridge:" + str(len(bridge)))

      # Delay for a while, so that the bridge disappears slowly
      time.sleep(0.25)

  elif b != block.GOLD_BLOCK.id:
    # There is treasure, and you are not already standing on gold
    # build another gold block below your player
    mc.setBlock(pos.x, pos.y-1, pos.z, block.GOLD_BLOCK.id)

    # Remember the coordinate of this new gold block...
    coordinate = [pos.x, pos.y-1, pos.z]
    # ...by adding it to the end of the bridge list
    bridge.append(coordinate)

    # You loose one point for each block of the bridge that is built
    score = score - 1

# Main game loop
while True:
  # Run the loop 10 times every second
  # It needs to run quite fast so that the program responds well
  time.sleep(0.1)

  # If there is no treasure placed yet,
  # and if a bridge has not yet been built
  if treasure_x == None and len(bridge) == 0:
    # Place a new treasure block randomly
    placeTreasure()

  # Perform any bridge-building activities
  buildBridge()

  # Perform any homing-beacon activitites
```

```
    homingBeacon()

    # Perform any hit-checking activities
    checkHit()

# END
```

5 Adventure 5: Using Data Files

5.1 Tip Chat

```python
# This program posts random hints and tips to the Minecraft chat.
# It reads those hits and tips from a separate text file.

# Import necessary modules
import mcpi.minecraft as minecraft
import time
import random

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# Set a constant for the filename, so that you can easily change it
# to read a different file later
FILENAME = "tips.txt"

# Open the file in read mode
f = open(FILENAME, "r")

# Read all lines from the file into a list called 'tips'
tips = f.readlines()

# Close the file when you have finished with it.
f.close()

# Game loop
while True:
  # Wait a random time between 3 and 7 seconds
  time.sleep(random.randint(3, 7))

  # Choose a random message from the 'tips' list
  msg = random.choice(tips)

  # Give the tip to the player. strip() strips out unwanted newlines
  mc.postToChat(msg.strip())

# END
```

5.2 CSV Build

```
# This program builds mazes in the Minecraft world.
# The maze data is stored in a CSV file.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# Define some constants for the different blocks you will use
# to build your maze out of. This allows you to experiment with
# different blocks and create interesting mazes
GAP   = block.AIR.id
WALL  = block.GOLD_BLOCK.id
FLOOR = block.GRASS.id

# This is the name of the file to read maze data from.
# It is a constant, so that you can change it easily to read from
# other data files in the future
FILENAME = "maze1.csv"

# Open the file containing your maze data
f = open(FILENAME, "r")

# Get the player position, and work out coordinates of the bottom corner
# The x and z are moved slightly to make sure that the maze doesn't get
# built on top of your players head
pos = mc.player.getTilePos()
ORIGIN_X = pos.x+1
ORIGIN_Y = pos.y
ORIGIN_Z = pos.z+1

# The z coordinate will vary at the end of each line of data
# So start it off at the origin of the maze
z = ORIGIN_Z

# Loop around every line of the file
for line in f.readlines():
  # Split the line into parts, wherever there is a comma
  data = line.split(",")

  # Restart the x coordinate every time round the "for line" loop
  x = ORIGIN_X

  # This is a nested loop (a loop inside another loop)
  # It loops through every item in the "data" list
  # It loops through cells, just like cells in a spreadsheet
  for cell in data:
    # Check if this cell of data is a gap or a wall
    if cell == "0": # f.readlines read it in as a string, so use quotes
      # choose the gap block id
      b = GAP
    else:
      # otherwise choose the wall block id
      b = WALL

    # build two layers of the required block id (held in the b variable)
    mc.setBlock(x, ORIGIN_Y, z, b)
    mc.setBlock(x, ORIGIN_Y+1, z, b)
    # build one layer of floor underneat it
    mc.setBlock(x, ORIGIN_Y-1, z, FLOOR)

    # move to the next x coordinate at the end of this "for cell" loop
    x = x + 1

  # move to the next z coordinate at the end of this "for line" loop
  z = z + 1

# END
```

5.3 Scan 3D

```
# This program scans a region of the Minecraft world and stores
# a representation of the object in a CSV file.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# Create some constants for the name of the file you want to scan to
# and the size of the object you want to scan
FILENAME = "tree.csv"
SIZEX = 5
SIZEY = 5
SIZEZ = 5

# Define a function that will scan at an x,y,z position to a file
# filename - name of the file to create
# originx,originy,originz - x,y,z coordinates of start of object to scan
def scan3D(filename, originx, originy, originz):
  # Open the file in write mode
  f = open(filename, "w")

  # Write the metadata line that has 3 sizes in it seprated by commas
  # the \n is a newline character (it presses ENTER for you at end)
  f.write(str(SIZEX) + "," + str(SIZEY) + "," + str(SIZEZ) + "\n")

  # Loop through all the y coordinates of the space to be scanned
  for y in range(SIZEY):
    # Write a blank line at the start of each layer of data
    f.write("\n")

    # Loop through all x coordinates of the space to be scanned
    for x in range(SIZEX):
      # Build up a "line" variable, it starts off empty
      line = ""

      # Loop through all z coordinates of the space to be scanned
      for z in range(SIZEZ):
        # get the block id at this x,y,z position
        blockid = mc.getBlock(originx+x, originy+y, originz+z)

        # If the "line" variable is empty, this is the first block
        # on this line, so it doesn't need a comma
        if line != "":
          # 'line' is not empty, so you must need a comma, so add it
          line = line + ","

        # Add the block id of this block to the end of 'line' variable
        line = line + str(blockid)

      # Note the indention. This is part of the 'for x' loop
      # write the whole line and press 'ENTER' at the end of it
      f.write(line + "\n")

  # Note the indentation. This is not part of any loop
  # Close the file when finished, so other programs can use it
  f.close()
```

```
# Get the position of the player
pos = mc.player.getTilePos()

# Scan an area into the file FILENAME
# It spans SIZEX,SIZEY,SIZEZ centered around the player,
# but note that it does not "dig down" under the player,
# which is why pos.y is used here
scan3D(FILENAME, pos.x-(SIZEX/2), pos.y, pos.z-(SIZEZ/2))

# END
```

5.4 Print 3D

```
# This program reads from a CSV file and builds blocks in the
# Minecraft world.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# Create a constant for the name of your data file
# This makes it easier to use a different file in the future
FILENAME = "object1.csv"

# Define a function that will print the contents of any file
# at any x,y,z coordinate in the Minecraft world
# filename - is the name of the file to open
# x,y,z - these are the coordinates of the origin (bottom corner)
def print3D(filename, originx, originy, originz):
  # Open the data file in read mode
  f = open(filename, "r")

  # Read all lines from the file into a 'lines' list variable
  lines = f.readlines()

  # The first line in the file is the metadata, it holds the 3 sizes
  # Split this line into parts, and set 3 variables, one for each size
  coords = lines[0].split(",")
  sizex = int(coords[0])
  sizey = int(coords[1])
  sizez = int(coords[2])

  # Use the 'lineidx' variable to keep track of the line number
  # in the loop below. Each time arount the loop, this will set to
  # the next line number, so that your loop processes one line at a time
  lineidx = 1

  # This first for loop reads through each vertical layer of the file
  for y in range(sizey):
    mc.postToChat(str(y)) # just so you can see what it is doing

    # Advance to the next line index
    lineidx = lineidx + 1

    # This inner loop reads through each east/west position in a line
    for x in range(sizex):
      # Get the whole line at this 'lineidx'
      line = lines[lineidx]

      # Advance to the next line index
      lineidx = lineidx + 1

      # Split this line into separate cells, at each comma
      data = line.split(",")

      # This (third) inner loop reads through each north/south position
      for z in range(sizez):
        # Get the blockid from the line at this z position
        # Convert it to a number by using int()
        blockid = int(data[z])
        # Build the block at the correct x,y,z position
        mc.setBlock(originx+x, originy+y, originz+z, blockid)

# Get the player position
pos = mc.player.getTilePos()

# "3D-print" the contents of the FILENAME file just to the side
print3D(FILENAME, pos.x+1, pos.y, pos.z+1)

# END
```

5.5 Duplicator

```
# This program builds a huge 3D duplicating machine.
# The machine can be used to scan and to print 3D objects all around
# the Minecraft world. Objects can be saved to a CSV file, loaded
# back into the duplicator room, and duplicated all over the world.

# Import necessary modules
import mcpi.minecraft as minecraft
import mcpi.block as block
import glob # needed to handle filename wildcards
import time
import random

# Connect to the Minecraft game
mc = minecraft.Minecraft.create()

# Set some constants for the size of the duplicator room
SIZEX = 10
SIZEY = 10
SIZEZ = 10

# Set a default location for the duplicator room
roomx = 1
roomy = 1
roomz = 1


# Define a function that builds the duplicator room
# It will build it at x,y,z
def buildRoom(x, y, z):
  # These 3 variables will have their value changed by this function
  # so they have to be listed as global here
  global roomx, roomy, roomz

  # Remember where the room has been built
  # so that it can be demolished later on
  roomx = x
  roomy = y
  roomz = z

  # Build the duplicator room out of glass
  mc.setBlocks(roomx, roomy, roomz,
    roomx+SIZEX+1, roomy+SIZEY+1, roomz+SIZEZ+1, block.GLASS.id)

  # Carve the insides and front face out with air
  mc.setBlocks(roomx+1, roomy+1, roomz,
    roomx+SIZEX, roomy+SIZEY, roomz+SIZEZ, block.AIR.id)


# Define a function that demolishes the room
def demolishRoom():
  # Set the whole area, including walls and contents, to air
  # Note how the roomx, roomy and roomz are used to remember
  # where the room was built in the first place
  mc.setBlocks(roomx, roomy, roomz,
    roomx+SIZEX+1, roomy+SIZEY+1, roomz+SIZEZ+1, block.AIR.id)


# Define a function to clean just the contents of the room
def cleanRoom():
  # Set the whole inside area to AIR
  mc.setBlocks(roomx+1, roomy+1, roomz+1,
    roomx+SIZEX, roomy+SIZEY, roomz+SIZEZ, block.AIR.id)


# Define a function that lists all CSV files
def listFiles():
  print("\nFILES:")

  # Get a list of files matching the wildcard *.csv into 'files'
  files = glob.glob("*.csv")

  # Loop through each filename in the list
  for filename in files:
    print(filename)

  # Print a blank line at the end
  # This is so that it separates the file list from the menu display
  print("\n")


# Define a function that will scan at an x,y,z position to a file
# filename - name of the file to create
# originx,originy,originz - x,y,z coordinates of start of object to scan
def scan3D(filename, originx, originy, originz):
  # Open the file in write mode
  f = open(filename, "w")

  # Write the metadata line that has 3 sizes in it seprated by commas
  # the \n is a newline character (it presses ENTER for you at end)
  f.write(str(SIZEX) + "," + str(SIZEY) + "," + str(SIZEZ) + "\n")

  # Loop through all the y coordinates of the space to be scanned
  for y in range(SIZEY):
    mc.postToChat("scan:" + str(y)) # so you can see it working

    # Write a blank line at the start of each layer of data
    f.write("\n")

    # Loop through all x coordinates of the space to be scanned
    for x in range(SIZEX):
      # Build up a "line" variable, it starts off empty
      line = ""

      # Loop through all z coordinates of the space to be scanned
      for z in range(SIZEZ):
        # get the block id at this x,y,z position
        blockid = mc.getBlock(originx+x, originy+y, originz+z)

        # If the "line" variable is empty, this is the first block
        # on this line, so it doesn't need a comma
        if line != "":
          # 'line' is not empty, so you must need a comma, so add it
          line = line + ","

        # Add block id of this block to the end of 'line' variable
        line = line + str(blockid)

      # Note the indention. This is part of the 'for x' loop
      # write the whole line and press 'ENTER' at the end of it
      f.write(line + "\n")

  # Note the indentation. This is not part of any loop
  # Close the file when finished, so other programs can use it
  f.close()
```

```
# Define a function that will print the contents of any file
# at any x,y,z coordinate in the Minecraft world
# filename - is the name of the file to open
# x,y,z - these are the coordinates of the origin (bottom corner)
def print3D(filename, originx, originy, originz):
  # Open the data file in read mode
  f = open(filename, "r")

  # Read all lines from the file into a 'lines' list variable
  lines = f.readlines()

  # The first line in the file is the metadata, it holds the 3 sizes
  # Split this line into parts, and set 3 variables, one for each size
  coords = lines[0].split(",")
  sizex = int(coords[0])
  sizey = int(coords[1])
  sizez = int(coords[2])

  # Use the 'lineidx' variable to keep track of the line number
  # in the loop below. Each time arount the loop, this will set to
  # the next line number, so that your loop processes one line at a time
  lineidx = 1

  # This first for loop scans through each vertical layer of the file
  for y in range(sizey):
    mc.postToChat("print:" + str(y)) # so you can see what it is doing

    # Advance to the next line index
    lineidx = lineidx + 1

    # This inner loop reads through each east/west position in a line
    for x in range(sizex):
      # Get the whole line at this 'lineidx'
      line = lines[lineidx]

      # Advance to the next line index
      lineidx = lineidx + 1

      # Split this line into separate cells, at each comma
      data = line.split(",")

      # This (third) inner loop reads through each north/south position
      for z in range(sizez):
        blockid = int(data[z])
        mc.setBlock(originx+x, originy+y, originz+z, blockid)


# Define a function that displays the menu
# This function will also ask the user for a choice number
# and it will check if the number in range or not.
# If it is not in range it will display the menu and try again
def menu():
  while True:
    print("DUPLICATOR MENU")
    print(" 1. BUILD the duplicator room")
    print(" 2. LIST files")
    print(" 3. SCAN from duplicator room to file")
    print(" 4. LOAD from file into duplicator room")
    print(" 5. PRINT from duplicator room to player.pos")
    print(" 6. CLEAN the duplicator room")
    print(" 7. DEMOLISH the duplicator room")
    print(" 8. QUIT")

    choice = int(input("please choose: "))
    if choice < 1 or choice > 8:
      print("Sorry, please choose a number between 1 and 8")
    else:
      return choice


# Game loop

# The anotherGo boolean variable decides if we want to go round
# the game loop another time
anotherGo = True

while anotherGo:
  # Display the menu and get a choice from the user
  choice = menu()

  if choice == 1: # BUILD
    # Build the room at the players position
    pos = mc.player.getTilePos()
    buildRoom(pos.x, pos.y, pos.z)

  elif choice == 2: # LIST
    # List all matching CSV files
    listFiles()

  elif choice == 3: # SCAN
    # Ask the user for a file to scan into
    filename = input("filename?")

    # Scan from the duplicator room into the file
    scan3D(filename, roomx+1, roomy+1, roomz+1)

  elif choice == 4: # LOAD
    # Ask the user for a file to load from
    filename = input("filename?")

    # Load the file into the duplicator room
    print3D(filename, roomx+1, roomy+1, roomz+1)

  elif choice == 5: # PRINT
    # Note, this is a really neat solution here, by reusing
    # the scan3D and print3D functions, and scanning via a temporary
    # file called 'scantemp' - it means you don't have to write
    # lots of extra printing and scanning code in order to offer this
    # option to your user

    # Scan the room into a temporary file called 'scantemp'
    scan3D("scantemp", roomx+1, roomy+1, roomz+1)

    # Get the position of the player
    pos = mc.player.getTilePos()

    # Print the contents of the temporary file 'scantemp' here
    print3D("scantemp", pos.x+1, pos.y, pos.z+1)

  elif choice == 6: # CLEAN
    # Clean out just the insides of the room
    cleanRoom()

  elif choice == 7: # DEMOLISH
```

```
    # Remove all trace of the room by demolishing it
    # This makes sure you don’t leave lots of "dead" rooms in your world
    demolishRoom()

  elif choice == 8: # QUIT
    # Finish the program
    # When anotherGo is False, the while loop (above) will finish
    anotherGo = False

# END
```

6 Adventure 6: Building 2D and 3D Structures

6.1 Lines Circles and Spheres

```
# This program shows how to create lines, spheres and circles in
# Minecraft using the MinecraftDrawing functions in minecraftstuff.


import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import time

#create the minecraft api
mc = minecraft.Minecraft.create()

#create the minecraft drawing object
mcdrawing = minecraftstuff.MinecraftDrawing(mc)

#get the players position
pos = mc.player.getTilePos()

#draw 3 lines
mcdrawing.drawLine(pos.x, pos.y, pos.z,
                   pos.x, pos.y+20, pos.z,
                   block.WOOL.id,1)
mcdrawing.drawLine(pos.x, pos.y, pos.z,
                   pos.x+20, pos.y, pos.z,
                   block.WOOL.id,2)
mcdrawing.drawLine(pos.x, pos.y, pos.z,
                   pos.x + 20, pos.y + 20, pos.z,
                   block.WOOL.id, 3)

#sleep so the player can move to a different position
time.sleep(5)

#draw a circle above the player
pos = mc.player.getTilePos()
mcdrawing.drawCircle(pos.x, pos.y + 20, pos.z, 20, block.WOOL.id, 4)
time.sleep(5)

#draw a sphere above the player
pos = mc.player.getTilePos()
mcdrawing.drawSphere(pos.x, pos.y + 20, pos.z, 15, block.WOOL.id, 5)
time.sleep(5)
```

6.2 Minecraft Clock

```
# This program uses the MinecraftDrawing functions in minecraftstuff
# to create a huge working clock in the sky.

import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import time
import datetime
import math

def findPointOnCircle(cx, cy, radius, angle):
    x = cx + math.sin(math.radians(angle)) * radius
    y = cy + math.cos(math.radians(angle)) * radius
    x = int(round(x, 0))
    y = int(round(y, 0))
    return(x,y)

#create connection to minecraft
mc = minecraft.Minecraft.create()

#create minecraft drawing object
mcdrawing = minecraftstuff.MinecraftDrawing(mc)

#get the players position
pos = mc.player.getTilePos()

#create clock face above the player
clockMiddle = pos
clockMiddle.y = clockMiddle.y + 25

CLOCK_RADIUS = 20
HOUR_HAND_LENGTH = 10
MIN_HAND_LENGTH = 18
SEC_HAND_LENGTH = 20

#use mc drawing object to draw the circle of the clock face
mcdrawing.drawCircle(clockMiddle.x, clockMiddle.y, clockMiddle.z,
                     CLOCK_RADIUS, block.DIAMOND_BLOCK.id)

#loop forever
while True:
    #Find the time
    # get the time
    timeNow = datetime.datetime.now()
    # hours
    hours = timeNow.hour
    if hours >= 12:
        hours = timeNow.hour - 12
    # minutes
    minutes = timeNow.minute
    # seconds
    seconds = timeNow.second

    #hour hand
    # what angle would a hour hand point to
    hourHandAngle = (360 / 12) * hours
    # use findPointOnCircle function to find the angle
    hourHandX, hourHandY = findPointOnCircle(clockMiddle.x, clockMiddle.y,
                                             HOUR_HAND_LENGTH, hourHandAngle)
    # draw hour hand
    mcdrawing.drawLine(clockMiddle.x, clockMiddle.y, clockMiddle.z,
                         hourHandX, hourHandY, clockMiddle.z,
                         block.DIRT.id)

    #minute hand
    # what angle would a minute hand point to
    minHandAngle = (360 / 60) * minutes
    # use findPointOnCircle function to find the angle
    minHandX, minHandY = findPointOnCircle(clockMiddle.x, clockMiddle.y,
                                           MIN_HAND_LENGTH, minHandAngle)
    # draw minute hand
    mcdrawing.drawLine(clockMiddle.x, clockMiddle.y, clockMiddle.z - 1,
                       minHandX, minHandY, clockMiddle.z-1,
                       block.WOOD_PLANKS.id)

    #second hand
    # what angle would a second hand point to
    secHandAngle = (360 / 60) * seconds
    # use findPointOnCircle function to find the angle
    secHandX, secHandY = findPointOnCircle(clockMiddle.x, clockMiddle.y,
                                           SEC_HAND_LENGTH, secHandAngle)
    # draw second hand
    mcdrawing.drawLine(clockMiddle.x, clockMiddle.y, clockMiddle.z+1,
                       secHandX, secHandY, clockMiddle.z+1,
                       block.STONE.id)

    #wait for 1 second
    time.sleep(1)

    #clear the time by drawing over the hands with AIR
    mcdrawing.drawLine(clockMiddle.x, clockMiddle.y, clockMiddle.z,
                       hourHandX, hourHandY, clockMiddle.z,
                       block.AIR.id)
    mcdrawing.drawLine(clockMiddle.x, clockMiddle.y, clockMiddle.z-1,
                       minHandX, minHandY, clockMiddle.z-1,
                       block.AIR.id)
    mcdrawing.drawLine(clockMiddle.x, clockMiddle.y, clockMiddle.z+1,
                       secHandX, secHandY, clockMiddle.z+1,
                       block.AIR.id)
```

6.3 Polygon

```
# This program shows you how to create polygons using the MinecraftDrawing
# functions in minecraftstuff.

import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import time

#create connection to minecraft
mc = minecraft.Minecraft.create()

#create minecraftstuff drawing object
mcdrawing = minecraftstuff.MinecraftDrawing(mc)

#get the players position
pos = mc.player.getTilePos()

#draw 2d shapes
# draw a triangle
points = minecraftstuff.Points()
points.add(pos.x, pos.y, pos.z)
points.add(pos.x + 20, pos.y, pos.z)
points.add(pos.x + 10, pos.y + 20, pos.z)
mcdrawing.drawFace(points, True, block.WOOL.id, 6)

#move the position on a bit
pos.x = pos.x + 25

#draw a square
points = minecraftstuff.Points()
points.add(pos.x, pos.y, pos.z)
points.add(pos.x + 20, pos.y, pos.z)
points.add(pos.x + 20, pos.y + 20, pos.z)
points.add(pos.x, pos.y + 20, pos.z)
mcdrawing.drawFace(points, False, block.WOOL.id, 7)

#move the position on a bit
pos.x = pos.x + 25

#4 sided odd shape
points = minecraftstuff.Points()
points.add(pos.x, pos.y, pos.z)
points.add(pos.x + 15, pos.y, pos.z)
points.add(pos.x + 20, pos.y + 15, pos.z)
points.add(pos.x, pos.y + 20, pos.z)
mcdrawing.drawFace(points, True, block.WOOL.id, 8)

#move the position on a bit
pos.x = pos.x + 25

#5 sided odd shape
points = minecraftstuff.Points()
points.add(pos.x, pos.y, pos.z)
points.add(pos.x + 20, pos.y + 5, pos.z)
points.add(pos.x + 15, pos.y + 20, pos.z)
points.add(pos.x + 5, pos.y + 15, pos.z)
points.add(pos.x, pos.y + 5, pos.z)
mcdrawing.drawFace(points, False, block.WOOL.id, 9)
```

6.4 Minecraft Pyramids

```
# This program uses the MinecraftDrawing polgyon functions in
# to create a huge pyramids in Minecraft.

#import python modules
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import math

#find point on circle function
def findPointOnCircle(cx, cy, radius, angle):
    x = cx + math.sin(math.radians(angle)) * radius
    y = cy + math.cos(math.radians(angle)) * radius
    x = int(round(x, 0))
    y = int(round(y, 0))
    return(x,y)

#create connection to minecraft
mc = minecraft.Minecraft.create()

#create minecraft drawing object
mcdrawing = minecraftstuff.MinecraftDrawing(mc)

#pyramid variables
#the midde of the pyramid will be the position of the player
middle = mc.player.getTilePos()

RADIUS = 20
HEIGHT = 10
SIDES = 4

#loop through the number of sides, each side will become a triangle
for side in range(0,SIDES):
    #find the first point of the triangle
    point1Angle = int(round((360 / SIDES) * side,0))
    point1X, point1Z = findPointOnCircle(middle.x, middle.z,
                                         RADIUS, point1Angle)

    #find the second point of the triangle
    point2Angle = int(round((360 / SIDES) * (side + 1),0))
    point2X, point2Z = findPointOnCircle(middle.x, middle.z,
                                         RADIUS, point2Angle)

    #draw the triangle
    # create the triangle points
    points = minecraftstuff.Points()
    points.add(point1X, middle.y, point1Z)
    points.add(point2X, middle.y, point2Z)
    points.add(middle.x, middle.y + HEIGHT, middle.z)

    # draw the triangle
    mcdrawing.drawFace(points, True, block.SANDSTONE.id)
```

7 Adventure 7: Giving Blocks a Mind of Their Own

7.1 Block Friend

```
# This program will create a block friend who will follow you around and
# generally be quite annoying!


import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import math
import time

#function get the distance between 2 points
def distanceBetweenPoints(point1, point2):
    xd = point2.x - point1.x
    yd = point2.y - point1.y
    zd = point2.z - point1.z
    return math.sqrt((xd*xd) + (yd*yd) + (zd*zd))

#constants
# how far away the block has to be before he stops following
TOO_FAR_AWAY = 15

#create minecraft and minecraftstuff objects
mc = minecraft.Minecraft.create()
mcdrawing = minecraftstuff.MinecraftDrawing(mc)

#set the blocks mood
blockMood = "happy"

#create the block next to the player
friend = mc.player.getTilePos()
friend.x = friend.x + 5
# find out the height of the world at x, y, so the friend is on top
friend.y = mc.getHeight(friend.x, friend.z)
# create the block
mc.setBlock(friend.x, friend.y, friend.z, block.DIAMOND_BLOCK.id)
# say hello
mc.postToChat("<block> Hello friend")
# the friends target is where he currently is
target = friend.clone()

while True:

    #get players position
    pos = mc.player.getTilePos()
    distance = distanceBetweenPoints(pos, friend)
    #apply the rules to work out where the block should be doing next
    # am I happy?
    if blockMood == "happy":
        #where is the player?  Are they near enough to me or should I
        # move to them?
        if distance < TOO_FAR_AWAY:
            target = pos.clone()
        else:
            blockMood = "sad"
            mc.postToChat(
                "<block> Come back. You are too far away. I need a hug!")
    # am I sad?
    elif blockMood == "sad":
        #if Im sad, I'll wait for my friend to come close and give me a hug
        # before I'm happy again
        #print distance
        if distance <= 1:
            blockMood = "happy"
            mc.postToChat("<block> Awww thanks. Lets go.")

    #move block to the target
    if friend != target:
        #get the blocks in between block friend and player, by 'drawing'
        # an imaginary line
        line = mcdrawing.getLine(friend.x, friend.y, friend.z,
                                 target.x, target.y, target.z)
        #loop through the blocks in between the friend and the target
        # loop to the last but 1 block (:-1) otherwise the friend will be
        #  sitting on top of the player
        for nextBlock in line[:-1]:
            #move the block friend to the next block
            # clear the old block by making it air
            mc.setBlock(friend.x, friend.y, friend.z, block.AIR.id)
            # set the position of the block friend to be the next block in-line
            friend = nextBlock.clone()
            # get the height of the land at the new position
            friend.y = mc.getHeight(friend.x, friend.z)
            # draw the block friend in the new position
            mc.setBlock(friend.x, friend.y, friend.z, block.DIAMOND_BLOCK.id)
            # time to sleep between each block move
            time.sleep(0.25)
        # we have reached our target, so set the target to be friend's position
        target = friend.clone()

    #sleep for a little bit to give the computer a rest!
    time.sleep(0.25)
```

7.2 Random Number

```
# This program shows how to create a random number using python.

import random
randomNo = random.randint(1,10)
print(randomNo)
```

7.3 Random Number If

```
# This program shows how to use random numbers and probability to make
# different things happen.

import random
if random.randint(1,10) == 10:
    print("This happens about 1 time in 10")
else:
    print("This happens about 9 times out of 10")
```

7.4 Block Friend Random

```
# This program will create a block friend who will follow you around and
# randomly be quite annoying!

import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import math
import time
#import random module
import random

#function get the distance between 2 points
def distanceBetweenPoints(point1, point2):
    xd = point2.x - point1.x
    yd = point2.y - point1.y
    zd = point2.z - point1.z
    return math.sqrt((xd*xd) + (yd*yd) + (zd*zd))

#constants
# how far away the block has to be before he stops following
TOO_FAR_AWAY = 15

#create minecraft and minecraftstuff objects
mc = minecraft.Minecraft.create()
mcdrawing = minecraftstuff.MinecraftDrawing(mc)

#set the blocks mood
blockMood = "happy"

#create the block next to the player
friend = mc.player.getTilePos()
friend.x = friend.x + 5
# find out the height of the world at x, y, so the friend is on top
friend.y = mc.getHeight(friend.x, friend.z)
# create the block
mc.setBlock(friend.x, friend.y, friend.z, block.DIAMOND_BLOCK.id)
# say hello
mc.postToChat("<block> Hello friend")
# the friends target is where he currently is
target = friend.clone()

while True:

    #get players position
    pos = mc.player.getTilePos()
    distance = distanceBetweenPoints(pos, friend)
    #apply the rules to work out where the block should be doing next
    # am I happy?
    if blockMood == "happy":
        #where is the player?  Are they near enough to me or should I
        # move to them?
        if distance < TOO_FAR_AWAY:
            target = pos.clone()
        else:
            blockMood = "sad"
            mc.postToChat(
                "<block> Come back. You are too far away. I need a hug!")

    # am I sad?
    elif blockMood == "sad":
        #if Im sad, I'll wait for my friend to come close and give me a hug
        # before I'm happy again
        #print distance
        if distance <= 1:
            blockMood = "happy"
            mc.postToChat("<block> Awww thanks. Lets go.")
        if random.randint(1,100) == 100:
            blockMood = "hadenough"
            mc.postToChat("<block> Thats it. I have had enough.")

    #Challenge
    #elif blockMood == "hadenough":
    #    if random.randint(1,50) = 50:
    #        blockMood = "sad"
    #        mc.postToChat("<block> I forgive you, can I have that hug now?")

    #move block to the target
    if friend != target:
        #get the blocks in between block friend and player, by 'drawing'
        # an imaginary line
        line = mcdrawing.getLine(friend.x, friend.y, friend.z,
                                 target.x, target.y, target.z)
        #loop through the blocks in between the friend and the target
        # loop to the last but 1 block (:-1) otherwise the friend will be
        #  sitting on top of the player
        for nextBlock in line[:-1]:
            #move the block friend to the next block
            # clear the old block by making it air
            mc.setBlock(friend.x, friend.y, friend.z, block.AIR.id)
            # set the position of the block friend to be the next block in-line
            friend = nextBlock.clone()
            # get the height of the land at the new position
            friend.y = mc.getHeight(friend.x, friend.z)
            # draw the block friend in the new position
            mc.setBlock(friend.x, friend.y, friend.z, block.DIAMOND_BLOCK.id)
            # time to sleep between each block move
            time.sleep(0.25)
        # we have reached our target, so set the target to be friend's position
        target = friend.clone()

    #sleep for a little bit to give the computer a rest!
    time.sleep(0.25)
```

7.5 Wooden Horse

```
# This program shows how to use the MinecraftShape functions in
# minecraftstuff to create and move a wooden horse shape.

import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import time

#create minecraft and minecraftdrawing objects
mc = minecraft.Minecraft.create()

#set the horses position
horsePos = mc.player.getTilePos()
horsePos.z = horsePos.z + 1
horsePos.y = horsePos.y + 1

#create the horseShape
horseShape = minecraftstuff.MinecraftShape(mc, horsePos)

#create the wooden horse by setting blocks
horseShape.setBlock(0,0,0,block.WOOD_PLANKS.id)
horseShape.setBlock(-1,0,0,block.WOOD_PLANKS.id)
horseShape.setBlock(1,0,0,block.WOOD_PLANKS.id)
horseShape.setBlock(-1,-1,0,block.WOOD_PLANKS.id)
horseShape.setBlock(1,-1,0,block.WOOD_PLANKS.id)
horseShape.setBlock(1,1,0,block.WOOD_PLANKS.id)
horseShape.setBlock(2,1,0,block.WOOD_PLANKS.id)

#make the horse move
for count in range(0,10):
    time.sleep(1)
    horseShape.moveBy(1,0,0)

horseShape.clear()
```

7.6 Alien Invasion

```python
# This program will create an Alien Invasion in Minecraft, resistance is
# futile though as you will never escape.

import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import math
import time
import random

#function get the distance between 2 points
def distanceBetweenPoints(point1, point2):
    xd = point2.x - point1.x
    yd = point2.y - point1.y
    zd = point2.z - point1.z
    return math.sqrt((xd*xd) + (yd*yd) + (zd*zd))

#constants
# will go here!
HOVER_HEIGHT = 15
ALIEN_TAUNTS = ["<aliens>You cannot run forever",
                "<aliens>Resistance is useless",
                "<aliens>We only want to be friends"]

#create minecraft and minecraftdrawing objects
mc = minecraft.Minecraft.create()
mcdrawing = minecraftstuff.MinecraftDrawing(mc)

#set the aliens position
alienPos = mc.player.getTilePos()
alienPos.y = alienPos.y + 50

#set the flying saucer's mode
mode = "landing"

alienShape = minecraftstuff.MinecraftShape(mc, alienPos)

alienShape.setBlock(-1,0,0,block.WOOL.id, 5)
alienShape.setBlock(0,0,-1,block.WOOL.id, 5)
alienShape.setBlock(1,0,0,block.WOOL.id, 5)
alienShape.setBlock(0,0,1,block.WOOL.id, 5)
alienShape.setBlock(0,-1,0,block.GLOWSTONE_BLOCK.id)
alienShape.setBlock(0,1,0,block.GLOWSTONE_BLOCK.id)

#loop
while mode != "missionaccomplished":
    #get the players position
    playerPos = mc.player.getTilePos()

    # if its landing, set target to be above players head
    if mode == "landing":
        mc.postToChat("<aliens> We do not come in peace - please panic")
        alienTarget = playerPos.clone()
        alienTarget.y = alienTarget.y + HOVER_HEIGHT
        mode = "attack"
    elif mode == "attack":
        #check to see if the alien ship is above the player
        if alienPos.x == playerPos.x and alienPos.z == playerPos.z:
            #the alienship is above the player
            mc.postToChat("<aliens>We have you now!")
            # create a room
            mc.setBlocks(0,50,0,6,56,6,block.BEDROCK.id)
            mc.setBlocks(1,51,1,5,55,5,block.AIR.id)
            mc.setBlock(3,55,3,block.GLOWSTONE_BLOCK.id)

            #beam up player
            mc.player.setTilePos(3,51,5)
            time.sleep(10)
            mc.postToChat("<aliens>Not very interesting at all - send it back")
            time.sleep(2)

            #send the player back to the original position
            mc.player.setTilePos(
                playerPos.x, playerPos.y, playerPos.z)

            #clear the room
            mc.setBlocks(0,50,0,6,56,6,block.AIR.id)
            mode = "missionaccomplished"

        else:
            #the player got away
            mc.postToChat(ALIEN_TAUNTS[random.randint(0,len(ALIEN_TAUNTS)-1)])
            alienTarget = playerPos.clone()
            alienTarget.y = alienTarget.y + HOVER_HEIGHT

    #move the flying saucer towards its target
    if alienPos != alienTarget:

        #get the blocks in between block friend and player,
        # by 'drawing' an imaginary line
        line = mcdrawing.getLine(alienPos.x, alienPos.y, alienPos.z,
                                 alienTarget.x, alienTarget.y, alienTarget.z)

        #loop through the blocks in between the alien and the target
        for nextBlock in line:
            #move the block friend to the next block
            # move the alien shape to the new position
            alienShape.move(nextBlock.x, nextBlock.y, nextBlock.z)

            # time to sleep between each block move
            time.sleep(0.25)

        # we have reached our target, so set the target to be friend's position
        alienPos = alienTarget.clone()

#clear the alien ship
alienShape.clear()
```

8 Adventure 8: Building a Game Controller with a BBC micro:bit

8.1 Hello

```
# This program shows how to connect Python to your micro:bit
# using the bitio package

# Import the microbit module, it will auto-connect to the micro:bit
import microbit

# END
```

8.2 Button

```
# This program shows how to sense a button press on your micro:bit

# Import necessary modules
import time
import microbit

# Display a message in the Python shell window
print("press button A to test")

# Loop forever waiting for the A button to be pressed
while True:
    # Check if the A buton has been pressed
    if microbit.button_a.was_pressed():
        # Display a message and delay a while to prevent flood of messages
        print("Button A pressed")
        time.sleep(0.5)

# END
```

8.3 Button 2

```
# This program shows how to write to your micro:bit display
# when you press a button

# Import necessary modules
import time
import microbit

# Display a message in the Python shell window
print("press button A to test")

# Loop forever
while True:
    # Sense when the A button has been pressed on the micro:bit
    if microbit.button_a.was_pressed():
        # Display message in Python shell window
        print("Button A pressed")
        # Also display letter A on micro:bit display for half a second
        microbit.display.show("A")
        time.sleep(0.5)
        microbit.display.clear()

# END
```

8.4 House

```
# This program displays a house icon on your micro:bit display
# when you arrive home

# Import necesssary modules
import microbit
import mcpi.minecraft as minecraft
import mcpi.block as block
import time

# Create a connection to Minecraft
mc = minecraft.Minecraft.create()

# Set the location of your home.
# NOTE: Set these to sensible coordinates before running this code
HOME_X = 0
HOME_Y = 0
HOME_Z = 0

# Build a woollen carpet here so you know where to build your house
mc.setBlock(HOME_X, HOME_Y, HOME_Z, block.WOOL.id, 15)

# The game loop runs forever
while True:
    # Slow things down a bit, so you don't overload your computer
    time.sleep(1)
    # Work out where Steve is standing
    pos = mc.player.getTilePos()
    # If Steve is standing on the doormat
    if pos.x == HOME_X and pos.z == HOME_Z:
        # Display a standard house image from the micro:bit image library
        microbit.display.show(microbit.Image.HOUSE)
    else:
        # Display a ? so you know your micro:bit is still doing something
        microbit.display.show('?')

# END
```

8.5 List Standard Images

```
# This program shows how to find out the names of all built-in images
# on your micro:bit

import microbit

# Display a list of all image names in the Python shell window
print(microbit.Image.STD_IMAGE_NAMES)

# END
```

8.6 House 2

```
# This program shows how to design a custom house icon for your micro:bit

# Import necessary modules
import microbit
import mcpi.minecraft as minecraft
import mcpi.block as block
import time

# Create a connection to the Minecraft game
mc = minecraft.Minecraft.create()

# Define a custom image for your house icon
MYHOUSE = microbit.Image("00900:09090:90009:99999:09990")

# Set the location of your home.
# NOTE: Set these to sensible coordinates before running this code
HOME_X = 0
HOME_Y = 0
HOME_Z = 0

# Build a woollen carpet here so you know where to build your house
mc.setBlock(HOME_X, HOME_Y, HOME_Z, block.WOOL.id, 15)

# The game loop runs forever
while True:
    # Slow things down a bit, so you don't overload your computer
    time.sleep(1)
    # Work out where Steve is standing
    pos = mc.player.getTilePos()
    # If Steve is standing on the doormat
    if pos.x == HOME_X and pos.z == HOME_Z:
        # Display your custom house image on the micro:bit display
        microbit.display.show(MYHOUSE)
    else:
        # Display a ? so you know your micro:bit is still doing something
        microbit.display.show('?')

# END
```

8.7 Banana

```
# This program senses when you touch a banana connected to your micro:bit

import microbit
import time

BANANA = microbit.Image("00090:00090:00990:09900:99000")

while True:
    time.sleep(0.25)
    if microbit.pin0.is_touched():
        microbit.display.show(BANANA)
    else:
        microbit.display.show('?')

# END
```

8.8 Detonator

```
# This program sets off an explosion when you touch a banana connected
# to your micro:bit
# NOTE: You can also use an orange, if you don’t have a banana handy

# Import necessary modules
import microbit
import mcpi.minecraft as minecraft
import mcpi.block as block
import time

# Define a custom image that looks a bit like a banana
BANANA = microbit.Image("00090:00090:00990:09900:99000")

# Create a connection to the Minecraft game
mc = minecraft.Minecraft.create()

# Define a function that will set off an explosion at a position in Minecraft
def bomb(x, y, z):
    # TNT goes into the centre of the blast area
    mc.setBlock(x+1, y, z+1, block.TNT.id)

    # Count down 5,4,3,2,1 on the micro:bit display
    for t in range(6):
        microbit.display.show(str(5-t))
        time.sleep(1)

    # The explosion is simulated, by setting a range of blocks to AIR
    mc.postToChat("BANG!")
    mc.setBlocks(x-10, y-5, z-10, x+10, y+10, z+10, block.AIR.id)

# The game loop runs forever
while True:
    # Slow things down a bit so you don’t overload your computer
    time.sleep(0.1)
    # Only when you touch the P0 pin
    if microbit.pin0.is_touched():
        # Show your custom banana image on the micro:bit display
        microbit.display.show(BANANA)
        # Get Steve’s position and set of an explosion where he is standing
        pos = mc.player.getTilePos()
        bomb(pos.x, pos.y, pos.z)
    else:
        # Put something on the display so you know it is still working
        microbit.display.show(’?’)

# END
```

8.9 Ball Game

```
# This program is a complete mini-game involving rolling a ball on a table

# Import necessary modules
import microbit
import mcpi.minecraft as minecraft
import mcpi.block as block
import time
import random

# CONSTANTS
# Change these to vary the size/difficulty of the game
TABLE_WIDTH = 20
TABLE_DEPTH = 20
TREASURE_COUNT = 25
TABLE = block.STONE.id
BALL = block.CACTUS.id
TREASURE = block.GOLD_BLOCK.id

# Connect to the Minecraft game and
# let the player know that the micro:bit is connected
mc = minecraft.Minecraft.create()
mc.postToChat("BBC micro:bit joined the game")

# VARIABLES

table_x = None
table_y = None
table_z = None
ball_x = None
ball_y = None
ball_z = None
speed_x = 0
speed_z = 0
remaining = 1

# Build a complete game table
def build_table(x, y, z):
    global table_x, table_y, table_z
    # Put a small gap around the edge in case of nearby mountains
    GAP = 10
    mc.setBlocks(x-GAP, y, z-GAP, x+TABLE_WIDTH+GAP, y+GAP, z+TABLE_DEPTH+GAP,
                 block.AIR.id)
    # Build the table as a big block
    mc.setBlocks(x-1, y, z-1, x+TABLE_WIDTH+1, y+1, z+TABLE_DEPTH+1, TABLE)
    # Carve out the middle, this creates the wall around the table
    mc.setBlocks(x, y+1, z, x+TABLE_WIDTH, y+1, z+TABLE_DEPTH, block.AIR.id)
    # Remember where the table was built, for later
    table_x = x
    table_y = y
    table_z = z

# Place random treasure blocks on the table
def place_treasure():
    y = table_y
    for i in range(TREASURE_COUNT):
        # BUGFIX: Prevent block being re-created in same pos
        # which would make it impossible to complete the game
        while True:
            # Choose a random position for the next block
            x = random.randint(0, TABLE_WIDTH) + table_x
            z = random.randint(0, TABLE_DEPTH) + table_z
```

```
                # Is it already a treasure block?
                b = mc.getBlock(x,y,z)
                if b == TREASURE: continue # if so, loop round to try again
                # Place treasure at the new random location
                mc.setBlock(x, y, z, TREASURE)
                break # break out of loop to move to next treasure block
            # BUGFIX END

# Move the ball based on the present ball speed
def move_ball():
    new_x = ball_x - speed_x
    new_z = ball_z - speed_z
    if ball_x != new_x or ball_z != new_z: # prevent screen flicker
        # Don't draw the ball if it falls off the table for some reason
        if is_on_table(new_x, new_z):
            move_ball_to(new_x, ball_y, new_z)

# Move the ball to a specific position on the table
def move_ball_to(x, y, z):
    global ball_x, ball_y, ball_z
    # Clear the existing ball if it has already been built
    if ball_x is not None:
        mc.setBlock(ball_x, ball_y, ball_z, block.AIR.id)
    # Update our record of where the ball is now
    ball_x = x
    ball_y = y
    ball_z = z
    # Build a block where the ball needs to be
    mc.setBlock(ball_x, ball_y, ball_z, BALL)

# Calculate a new speed value, given the existing speed and amount of tilt.
# You can change this algorithm to make the acceleration and deceleration of
# the ball more life-like
def new_speed(speed, tilt):
    # Create a dead spot when the micro:bit is flat to slow/stop the ball
    if abs(tilt) < 300: tilt = 0
    # Reduce the range of the tilt to roughly -3..+3
    tilt = tilt / 300
    # Adjust the speed gradually each time, to simulate simple acceleration
    if tilt < speed:
        speed = speed - 1
    elif tilt > speed:
        speed = speed + 1
    return speed

# Sense the amount of tilt of the micro:bit and update the speed variables
def check_tilt():
    global speed_x, speed_z
    speed_x = new_speed(speed_x, microbit.accelerometer.get_x())
    speed_z = new_speed(speed_z, microbit.accelerometer.get_y())
    #print(speed_x, speed_z)

# Work out if a given ball position is on the table or not.
# This can be used to prevent the ball rolling off the table.
# It can also be used to sense if it rolls off the table.
def is_on_table(x, z):
    if x < table_x or x > table_x + TABLE_WIDTH:
        return False
    if z < table_z or z > table_z + TABLE_DEPTH:
        return False
    return True

# Check what is below the ball, and action it accordingly
```

```
def check_below():
    global remaining
    # What is below the ball?
    block_below = mc.getBlock(ball_x, ball_y-1, ball_z)
    if block_below == TREASURE:
        # It's treasure, so collect it and update the count of remaining items
        mc.setBlock(ball_x, ball_y-1, ball_z, block.AIR.id)
        remaining = remaining - 1
        microbit.display.show(remaining)
    elif block_below == block.AIR.id:
        # It's a hole, so introduce a time-penalty
        move_ball_to(ball_x, ball_y-1, ball_z)
        while not microbit.button_b.was_pressed():
            time.sleep(0.1)
        # Bounce the ball out to a random new position
        move_ball_to(table_x + random.randint(0, TABLE_WIDTH), table_y+1,
                     table_z + random.randint(0, TABLE_DEPTH))

# Wait for the user to start the game
def wait_for_start():
    # A helpful message so that the user knows what to do
    mc.postToChat("press B to start")
    microbit.display.show("?")
    # Wait for the B button to be pressed on the micro:bit
    while not microbit.button_b.was_pressed():
        time.sleep(0.1)

    # Countdown 5,4,3,2,1
    for t in range(6):
        microbit.display.show(str(5-t))
        time.sleep(0.5)

# Build the complete game with table and treasure
def build_game():
    global speed_x, speed_z, remaining

    # Get Steve's position, so the table can be built nearby
    pos = mc.player.getTilePos()
    # Build the table close by
    build_table(pos.x, pos.y-2, pos.z)
    # Start the ball at a random position
    move_ball_to(table_x + random.randint(0, TABLE_WIDTH), table_y+1,
                 table_z + random.randint(0, TABLE_DEPTH))
    # Place all the treasure at random locations
    place_treasure()
    # Place Steve at a good viewing point
    mc.player.setTilePos(table_x + TABLE_WIDTH/2, table_y+1, table_z)
    # Start off the game variables at sensible values
    remaining = TREASURE_COUNT
    speed_x = 0
    speed_z = 0

# Play one complete game from start to end
def play_game():
    # Remember what time the game was started
    start_time = time.time()
    # Loop around a game loop until all treasure is collected
    while remaining > 0:
        # Slow things down a bit, this loop runs 10 times per second
        time.sleep(0.1)
        # Find out how much the user is tilting the micro:bit
        check_tilt()
        # Move the ball based on the tilt
        move_ball()
        # Process treasure or air blocks below the ball
        check_below()
    # Remember what time the game finished
    end_time = time.time()
    # Display the total game time taken to collect all treasure
    mc.postToChat("game time=" + str(int(end_time-start_time)))

# The main program.
# Loops forever, until you press CTRL-C
while True:
    wait_for_start()
    build_game()
    play_game()

# END
```

9 Adventure 9: The Big Adventure: Crafty Crossing

9.1 Crafty Crossing

```
# This program is a full mini game inside Minecraft called "crafty crossing"
# where you have to get to the other side of an arena while avoiding obstacles
# and collecting diamonds.

#import modules
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import time
import random
import threading

#arena constants
ARENAX = 10
ARENAZ = 20
ARENAY = 3

#Create the Arena
def createArena(pos):
    #create connection to minecraft
    mc = minecraft.Minecraft.create()

    #Create the floor
    mc.setBlocks(pos.x - 1 , pos.y, pos.z - 1,
                 pos.x + ARENAX + 1, pos.y - 3, pos.z + ARENAZ + 1,
                 block.GRASS.id)

    #Create the walls on the outside of the arena
    mc.setBlocks(pos.x - 1, pos.y + 1, pos.z - 1,
                 pos.x + ARENAX + 1, pos.y + ARENAY, pos.z + ARENAZ + 1,
                 block.GLASS.id)
    mc.setBlocks(pos.x, pos.y + 1, pos.z,
                 pos.x + ARENAX, pos.y + ARENAY, pos.z + ARENAZ,
                 block.AIR.id)

#Build the obstacles
#The Wall
def theWall(arenaPos, wallZPos):
    #create connection to minecraft
    mc = minecraft.Minecraft.create()

    #create the wall shape
    wallPos = minecraft.Vec3(arenaPos.x, arenaPos.y + 1, arenaPos.z + wallZPos)
    wallShape = minecraftstuff.MinecraftShape(mc, wallPos)

    #create the wall
    wallShape.setBlocks(
        0, 1, 0,
        ARENAX, ARENAY - 1, 0,
        block.BRICK_BLOCK.id)

    #move the wall up and down
    while not gameOver:
        wallShape.moveBy(0,1,0)
        time.sleep(1)
        wallShape.moveBy(0,-1,0)
        time.sleep(1)

#The River
```

```
def theRiver(arenaPos, riverZPos):
    #create connection to minecraft
    mc = minecraft.Minecraft.create()

    #constants
    RIVERWIDTH = 4
    BRIDGEWIDTH = 2

    #create the river
    mc.setBlocks(arenaPos.x,
                 arenaPos.y - 2,
                 arenaPos.z + riverZPos,
                 arenaPos.x + ARENAX,
                 arenaPos.y,
                 arenaPos.z + riverZPos + RIVERWIDTH - 1,
                 block.AIR.id)
    #fill with water
    mc.setBlocks(arenaPos.x,
                 arenaPos.y - 2,
                 arenaPos.z + riverZPos,
                 arenaPos.x + ARENAX,
                 arenaPos.y - 2,
                 arenaPos.z + riverZPos + RIVERWIDTH - 1,
                 block.WATER.id)
    #create the bridge shape
    bridgePos = minecraft.Vec3(arenaPos.x,
                               arenaPos.y,
                               arenaPos.z + riverZPos + 1)
    bridgeShape = minecraftstuff.MinecraftShape(mc, bridgePos)

    bridgeShape.setBlocks(
        0, 0, 0,
        BRIDGEWIDTH - 1, 0, RIVERWIDTH - 3,
        block.WOOD_PLANKS.id)

    #move the bridge left and right
    #how many steps are there between the left and right side of the arena
    steps = ARENAX - BRIDGEWIDTH + 1
    while not gameOver:
        for left in range(0, steps):
            bridgeShape.moveBy(1,0,0)
            time.sleep(1)
        for right in range(0, steps):
            bridgeShape.moveBy(-1,0,0)
            time.sleep(1)

def theHoles(arenaPos, holesZPos):
    #create connection to minecraft
    mc = minecraft.Minecraft.create()

    #constants
    HOLES = 15
    HOLESWIDTH = 3

    while not gameOver:
        #create random holes which open up for a few seconds, close,
        # then change to a different set of holes
        holes = []
        #find some random holes
        for count in range(0,HOLES):
            x = random.randint(arenaPos.x,
                               arenaPos.x + ARENAX)
            z = random.randint(arenaPos.z + holesZPos,
                                arenaPos.z + holesZPos + HOLESWIDTH)
            holes.append(minecraft.Vec3(x, arenaPos.y, z))
        #turn the holes black before opening them up
        for hole in holes:
            mc.setBlock(hole.x, hole.y, hole.z, block.WOOL.id, 15)
        time.sleep(0.25)
        #open up the holes
        for hole in holes:
            mc.setBlocks(hole.x, hole.y, hole.z,
                         hole.x, hole.y - 2, hole.z,
                         block.AIR.id)
        time.sleep(2)
        #close up the holes
        for hole in holes:
            mc.setBlocks(hole.x, hole.y, hole.z,
                         hole.x, hole.y - 2, hole.z,
                         block.GRASS.id)
        time.sleep(0.25)

#Place the diamonds
def createDiamonds(arenaPos, number):
    #create connection to minecraft
    mc = minecraft.Minecraft.create()

    #create the number of diamonds required
    for diamond in range(0, number):
        #create a random position
        x = random.randint(arenaPos.x, arenaPos.x + ARENAX)
        z = random.randint(arenaPos.z, arenaPos.z + ARENAZ)
        #create the diamond block
        mc.setBlock(x, arenaPos.y + 1, z, block.DIAMOND_BLOCK.id)

#Main program
#create minecraft object
mc = minecraft.Minecraft.create()

#create the gameOver flag
gameOver = False

#arena pos
arenaPos = mc.player.getTilePos()

#build the arena
createArena(arenaPos)

#create the wall
WALLZ = 10
#theWall(arenaPos, WALLZ)
wall_t = threading.Thread(
    target = theWall,
    args = (arenaPos, WALLZ))
wall_t.start()

#create the river
RIVERZ = 4
#theRiver(arenaPos, RIVERZ)
river_t = threading.Thread(
    target = theRiver,
    args = (arenaPos, RIVERZ))
river_t.start()

#create the holes
HOLESZ = 15
```

```
#theHoles(arenaPos, HOLESZ)
holes_t = threading.Thread(
    target = theHoles,
    args = (arenaPos, HOLESZ))
holes_t.start()

#level constants
LEVELS = 3
DIAMONDS = [3,5,9]
TIMEOUTS = [30,25,20]

#set level and points
level = 0
points = 0

#game loop, while not game over
while not gameOver:
    #create the diamonds
    createDiamonds(arenaPos, DIAMONDS[level])
    diamondsLeft = DIAMONDS[level]

    #position the player at the start of the arena
    mc.player.setPos(arenaPos.x + 1, arenaPos.y + 1, arenaPos.z + 1)

    #start the clock
    start = time.time()

    #set the level complete flag
    levelComplete = False
    #level loop
    while not gameOver and not levelComplete:
        #sleep for a bit
        time.sleep(0.1)

        #has player hit a diamond?
        hits = mc.events.pollBlockHits()
        for hit in hits:
            blockHitType = mc.getBlock(hit.pos.x, hit.pos.y, hit.pos.z)
            if blockHitType == block.DIAMOND_BLOCK.id:
                #turn the block to AIR
                mc.setBlock(hit.pos.x,hit.pos.y, hit.pos.z, block.AIR.id)
                #reduce the diamonds to collect by 1
                diamondsLeft = diamondsLeft - 1

        #get the players position
        pos = mc.player.getTilePos()

        #has player fallen down
        if pos.y < arenaPos.y:
            #put them back to the start
            mc.player.setPos(arenaPos.x + 1, arenaPos.y + 1, arenaPos.z + 1)

        #has the player got to the end of the arena and got all the diamonds?
        if pos.z == arenaPos.z + ARENAZ and diamondsLeft == 0:
            levelComplete = True

        #has the time expired
        secondsLeft = TIMEOUTS[level] - (time.time() - start)
        if secondsLeft < 0:
            gameOver = True
            mc.postToChat("Out of time...")

    #level complete?
    if levelComplete:
        #calculate points
        #1 points for every diamond
        #1 x multipler for every second left on the clock
        points = points + (DIAMONDS[level] * int(secondsLeft))
        mc.postToChat("Level Complete - Points = " + str(points))
        #set it to the next level
        level = level + 1
        #if its the last level, set it to gameOver
        if level == LEVELS:
            gameOver = True
            mc.postToChat("Congratulations - All levels complete")

#its game over
mc.postToChat("Game Over - Points = " + str(points))
```

9.2 Crafty Crossing With BBC micro:bit

```
# This program is a full mini game inside Minecraft called "crafty crossing"
# where you have to get to the other side of an arena while avoiding obstacles
# and collecting diamonds. The micro:bit is also used to display how many
# diamonds are left to collect and when time is running out.

#import modules
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import time
import random
import threading

#microbit - start
import microbit
#microbit - end

#arena constants
ARENAX = 10
ARENAZ = 20
ARENAY = 3

#Create the Arena
def createArena(pos):
    #create connection to minecraft
    mc = minecraft.Minecraft.create()

    #Create the floor
    mc.setBlocks(pos.x - 1 , pos.y, pos.z - 1,
                 pos.x + ARENAX + 1, pos.y - 3, pos.z + ARENAZ + 1,
                 block.GRASS.id)

    #Create the walls on the outside of the arena
    mc.setBlocks(pos.x - 1, pos.y + 1, pos.z - 1,
                 pos.x + ARENAX + 1, pos.y + ARENAY, pos.z + ARENAZ + 1,
                 block.GLASS.id)
    mc.setBlocks(pos.x, pos.y + 1, pos.z,
                 pos.x + ARENAX, pos.y + ARENAY, pos.z + ARENAZ,
                 block.AIR.id)

#Build the obstacles
#The Wall
def theWall(arenaPos, wallZPos):
    #create connection to minecraft
    mc = minecraft.Minecraft.create()

    #create the wall shape
    wallPos = minecraft.Vec3(arenaPos.x, arenaPos.y + 1, arenaPos.z + wallZPos)
    wallShape = minecraftstuff.MinecraftShape(mc, wallPos)

    #create the wall
    wallShape.setBlocks(
        0, 1, 0,
        ARENAX, ARENAY - 1, 0,
        block.BRICK_BLOCK.id)

    #move the wall up and down
    while not gameOver:
        wallShape.moveBy(0,1,0)
        time.sleep(1)
        wallShape.moveBy(0,-1,0)
        time.sleep(1)

#The River
def theRiver(arenaPos, riverZPos):
    #create connection to minecraft
    mc = minecraft.Minecraft.create()

    #constants
    RIVERWIDTH = 4
    BRIDGEWIDTH = 2

    #create the river
    mc.setBlocks(arenaPos.x,
                 arenaPos.y - 2,
                 arenaPos.z + riverZPos,
                 arenaPos.x + ARENAX,
                 arenaPos.y,
                 arenaPos.z + riverZPos + RIVERWIDTH - 1,
                 block.AIR.id)
    #fill with water
    mc.setBlocks(arenaPos.x,
                 arenaPos.y - 2,
                 arenaPos.z + riverZPos,
                 arenaPos.x + ARENAX,
                 arenaPos.y - 2,
                 arenaPos.z + riverZPos + RIVERWIDTH - 1,
                 block.WATER.id)
    #create the bridge shape
    bridgePos = minecraft.Vec3(arenaPos.x,
                               arenaPos.y,
                               arenaPos.z + riverZPos + 1)
    bridgeShape = minecraftstuff.MinecraftShape(mc, bridgePos)

    bridgeShape.setBlocks(
        0, 0, 0,
        BRIDGEWIDTH - 1, 0, RIVERWIDTH - 3,
        block.WOOD_PLANKS.id)

    #move the bridge left and right
    #how many steps are there between the left and right side of the arena
    steps = ARENAX - BRIDGEWIDTH + 1
    while not gameOver:
        for left in range(0, steps):
            bridgeShape.moveBy(1,0,0)
            time.sleep(1)
        for right in range(0, steps):
            bridgeShape.moveBy(-1,0,0)
            time.sleep(1)

def theHoles(arenaPos, holesZPos):
    #create connection to minecraft
    mc = minecraft.Minecraft.create()

    #constants
    HOLES = 15
    HOLESWIDTH = 3

    while not gameOver:
        #create random holes which open up for a few seconds, close,
        # then change to a different set of holes
        holes = []
        #find some random holes
        for count in range(0,HOLES):
            x = random.randint(arenaPos.x,
                                   arenaPos.x + ARENAX)
            z = random.randint(arenaPos.z + holesZPos,
                                   arenaPos.z + holesZPos + HOLESWIDTH)
            holes.append(minecraft.Vec3(x, arenaPos.y, z))
        #turn the holes black before opening them up
        for hole in holes:
            mc.setBlock(hole.x, hole.y, hole.z, block.WOOL.id, 15)
        time.sleep(0.25)
        #open up the holes
        for hole in holes:
            mc.setBlocks(hole.x, hole.y, hole.z,
                         hole.x, hole.y - 2, hole.z,
                         block.AIR.id)
        time.sleep(2)
        #close up the holes
        for hole in holes:
            mc.setBlocks(hole.x, hole.y, hole.z,
                         hole.x, hole.y - 2, hole.z,
                         block.GRASS.id)
        time.sleep(0.25)

#Place the diamonds
def createDiamonds(arenaPos, number):
    #create connection to minecraft
    mc = minecraft.Minecraft.create()

    #create the number of diamonds required
    for diamond in range(0, number):
        #create a random position
        x = random.randint(arenaPos.x, arenaPos.x + ARENAX)
        z = random.randint(arenaPos.z, arenaPos.z + ARENAZ)
        #create the diamond block
        mc.setBlock(x, arenaPos.y + 1, z, block.DIAMOND_BLOCK.id)

#Main program
#create minecraft object
mc = minecraft.Minecraft.create()

#create the gameOver flag
gameOver = False

#arena pos
arenaPos = mc.player.getTilePos()

#build the arena
createArena(arenaPos)

#create the wall
WALLZ = 10
#theWall(arenaPos, WALLZ)
wall_t = threading.Thread(
    target = theWall,
    args = (arenaPos, WALLZ))
wall_t.start()

#create the river
RIVERZ = 4
#theRiver(arenaPos, RIVERZ)
river_t = threading.Thread(
    target = theRiver,
    args = (arenaPos, RIVERZ))
river_t.start()
```

```
#create the holes
HOLESZ = 15
#theHoles(arenaPos, HOLESZ)
holes_t = threading.Thread(
    target = theHoles,
    args = (arenaPos, HOLESZ))
holes_t.start()

#level constants
LEVELS = 3
DIAMONDS = [3,5,9]
TIMEOUTS = [30,25,20]

#set level and points
level = 0
points = 0

#microbit - start
#wait for the button to be pressed
mc.postToChat("Press button A to start")
microbit.display.scroll("A to start")
while not microbit.button_a.was_pressed():
    time.sleep(0.1)
#microbit - end

#game loop, while not game over
while not gameOver:
    #create the diamonds
    createDiamonds(arenaPos, DIAMONDS[level])
    diamondsLeft = DIAMONDS[level]

    #microbit - start
    #wait for the button to be pressed
    microbit.display.show(diamondsLeft)
    #microbit - end

    #position the player at the start of the arena
    mc.player.setPos(arenaPos.x + 1, arenaPos.y + 1, arenaPos.z + 1)

    #start the clock
    start = time.time()

    #set the level complete flag
    levelComplete = False
    #level loop
    while not gameOver and not levelComplete:
        #sleep for a bit
        time.sleep(0.1)

        #has player hit a diamond?
        hits = mc.events.pollBlockHits()
        for hit in hits:
            blockHitType = mc.getBlock(hit.pos.x, hit.pos.y, hit.pos.z)
            if blockHitType == block.DIAMOND_BLOCK.id:
                #turn the block to AIR
                mc.setBlock(hit.pos.x,hit.pos.y, hit.pos.z, block.AIR.id)
                #reduce the diamonds to collect by 1
                diamondsLeft = diamondsLeft - 1

        #get the players position
        pos = mc.player.getTilePos()
        #has player fallen down
        if pos.y < arenaPos.y:
            #put them back to the start
            mc.player.setPos(arenaPos.x + 1, arenaPos.y + 1, arenaPos.z + 1)

        #has the player got to the end of the arena and got all the diamonds?
        if pos.z == arenaPos.z + ARENAZ and diamondsLeft == 0:
            levelComplete = True

        #has the time expired
        secondsLeft = TIMEOUTS[level] - (time.time() - start)

        #microbit - start
        if int(secondsLeft) < 6:
            microbit.display.show(
                microbit.Image.ALL_CLOCKS[int(secondsLeft)])
        else:
            microbit.display.show(diamondsLeft)
        #microbit - end

        if secondsLeft < 0:
            #microbit - start
            #microbit.display.show(
            #    microbit.Image.CLOCK12)
            #microbit - end
            gameOver = True
            mc.postToChat("Out of time...")

    #level complete?
    if levelComplete:
        #calculate points
        #1 points for every diamond
        #1 x multipler for every second left on the clock
        points = points + (DIAMONDS[level] * int(secondsLeft))
        mc.postToChat("Level Complete - Points = " + str(points))
        #set it to the next level
        level = level + 1
        #if its the last level, set it to gameOver
        if level == LEVELS:
            gameOver = True
            mc.postToChat("Congratulations - All levels complete")

#its game over
mc.postToChat("Game Over - Points = " + str(points))
```