MANGO RUN

GAME DESIGN DOCUMENT

LE BAZAAR DE LEO

2023

SUMMARY

CONCEPT 2

GAMEPLAY OVERVIEW 2

CONTROLS 2

CHARACTERS 2

LEVEL DESIGN 3

SCORING SYSTEM 3

SOUND AND MUSIC 3

ART AND VISUALS 3

USER INTERFACE (UI) 3

PLATFORMS 3

CLOSING REMARKS 4

SCREENSHOTS 5

CODE 5

CONCEPT

"Mango Run" is an action-adventure game inspired by the dinosaur Chrome game, adapted for the Thumby micro console from Tinycircuits. The player will control a Skeleton character, guiding it through an endless level filled with enemies, plants, and obstacles. The objective is to survive as long as possible by skillfully jumping over the obstacles while collecting mangoes to earn points.

GAMEPLAY OVERVIEW

- The game is an endless runner with 2D side-scrolling mechanics.
- The Skeleton character runs automatically from left to right, and the player must control its jumps to avoid obstacles.
- The player earns points by collecting mangoes placed throughout the levels.
- The game ends if the Skeleton collides with an enemy, plant, or obstacle.

CONTROLS

- The Thumby micro console's buttons will be used for controls.
- The 'A' button will make the Skeleton jump.

CHARACTERS

- Skeleton (Playable Character): A nimble and agile character that the player controls.
- Carnivorous Plants: Stationary obstacles that can snap at the Skeleton if it comes too close.
- Monsters: enemies that the Skeleton must avoid or jump over.

LEVEL DESIGN

- The game will have a procedurally generated level increasing in difficulty. As the player progresses, the speed and obstacles improves.
- Obstacles, enemies, and mango placements will vary in each level to keep the gameplay fresh.

SCORING SYSTEM

- Collecting mangoes awards points to the player.
- The longer the player survives, the higher the score they achieve.

SOUND AND MUSIC

- Catchy and upbeat background music will accompany the gameplay.
- Sound effects will enhance the player's experience, such as jump sounds, mango collection sounds, and enemy interactions.

ART AND VISUALS

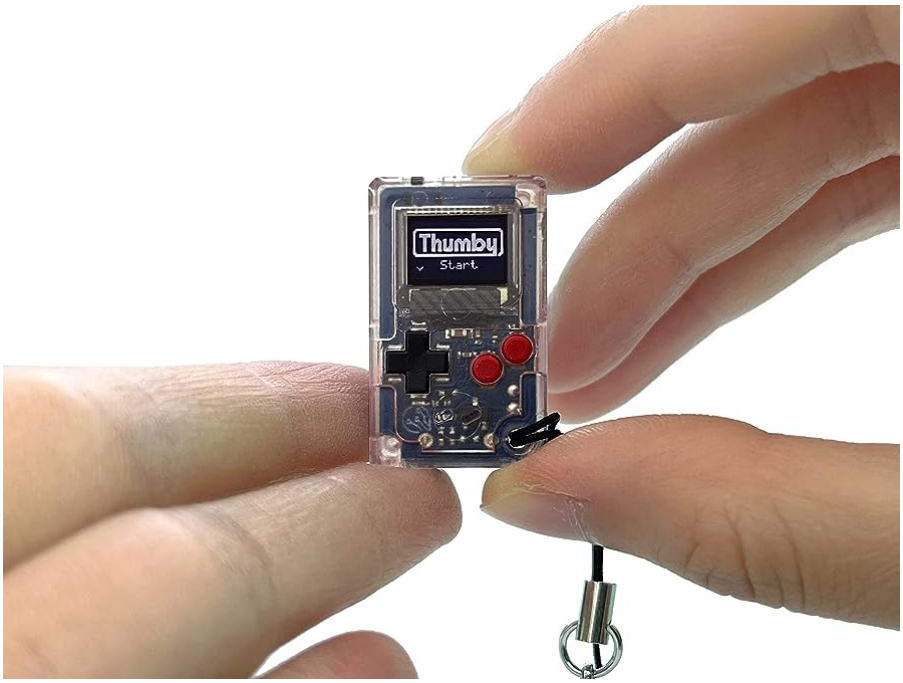
- Pixel art graphics will be used to give the game a retro and nostalgic feel.
- Only black and white colors will be used as Thumby only support Bitmap sprites which will enhance the retro visual feeling and appeal.

USER INTERFACE (UI)

- The UI will be simple and intuitive, displaying the player's score.

PLATFORMS

- Initially, the game will be developed for the Thumby micro console from Tinycircuits.



- Depending on its success, the game can be expanded to other platforms, such as mobile devices or PCs. But I highly doubt it. I will make a lot of copies and gift for family and friends though.

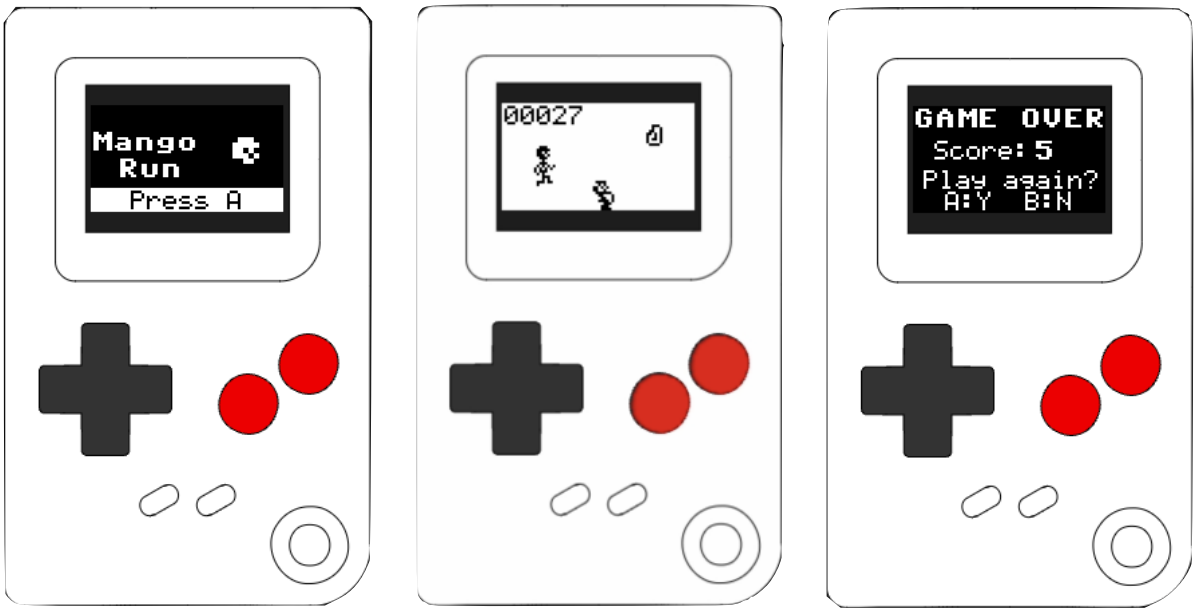
CLOSING REMARKS

"Mango Run" is my first ever game made in my game developer journey. I wanted to start small, indeed very small.

My aim is to provide players with an engaging and challenging experience while paying homage to the classic dinosaur Chrome game and learning to code simple yet extremely efficient code. The game is written in micro Python code.

With its intuitive controls, charming pixel art, and endless fun, I hope to captivate players of all ages and deliver an exciting gaming adventure.

SCREENSHOTS



CODE

```
# Mango Run
#
# A run and dodge game inspired by a famous dinosaur that can be seen when no Internet is available.
# Created by Le Bazaar de Leo.
#
# Some parts taken from TinySaur by Mason Watmough for TinyCircuits

'''
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
```

```
    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
'''

import ssd1306
import machine
import time
import random
import gc
import utime
import thumby

machine.freq(125000000)
gc.enable() # This line helps make sure we don't run out of memory

from framebuf import FrameBuffer, MONO_VLSB # Graphics stuff

# Interesting game parameters
XVel = 0.06
YVel = 0
YPos = 0
Gravity = 0.15
MaxFPS = 60
Points = 0
GameRunning = True
EnemyPos = random.randint(72, 300)
CloudPos = random.randint(60, 200)
CoinPos = random.randint(60, 200)
JumpSoundTimer = 0

def fscore(s):
    return '{:0>{w}}'.format(s, w=5)

# Sprite data


# BITMAP: width: 22, height: 14
titleFace = bytearray([0,0,0,0,0,0,240,240,252,252,60,60,252,252,48,48,0,0,0,0,0,0,
        0,0,0,0,0,0,3,3,3,3,15,15,12,12,3,3,0,0,0,0,0,0])

# BITMAP: width: 16, height: 16
```

```
PlayerRunFrame1 =
bytearray([255,255,255,255,255,227,65,9,81,235,255,255,255,255,255,255,
        255,255,255,255,249,54,185,130,233,142,189,254,255,255,255,255])

# BITMAP: width: 16, height: 16
PlayerRunFrame2 =
bytearray([255,255,255,255,255,227,65,9,81,235,255,255,255,255,255,255,
        255,255,255,255,187,92,233,2,121,254,253,255,255,255,255,255])

# BITMAP: width: 16, height: 16
PlayerRunFrame3 =
bytearray([255,255,255,255,255,99,65,9,81,235,255,255,255,255,255,255,
        255,255,255,253,186,95,232,2,121,254,253,255,255,255,255,255])

# Sky decorations: BigCloud, Sun, LittleCloud
# BITMAP: width: 16, height: 8
SkySets = [bytearray([159,79,99,89,189,115,115,101,92,126,126,81,87,79,31,191]),
        bytearray([255,247,190,213,107,156,221,92,221,156,107,213,190,247,255,255]),
        bytearray([255,255,255,255,255,145,110,126,106,222,106,126,110,145,255,255])]
CloudSpr = SkySets[0]

# Enemies: Flower, Ghost, Mushroom, Hedgehog, Ghost2, Toad
# BITMAP: width: 8, height: 12
EnemySets = [[bytearray([115,65,202,31,31,202,224,243,14,14,12,0,0,9,12,14]),
bytearray([63,99,193,10,26,192,97,127,15,14,12,0,0,9,12,15])],
         [bytearray([1,115,87,119,87,119,1,51,8,0,1,4,13,8,0,3]),
bytearray([1,99,87,119,85,97,1,55,0,1,1,12,13,1,0,11])],
         [bytearray([143,131,25,0,0,25,131,143,8,0,1,7,15,7,0,9]),
bytearray([143,131,49,0,0,49,131,143,9,0,7,15,3,1,0,8])],
         [bytearray([63,135,27,135,91,135,91,7,14,2,0,2,9,2,1,0]),
bytearray([31,67,45,67,173,67,173,3,14,2,2,0,14,2,2,0])],
         [bytearray([62,1,146,241,145,242,1,62,7,0,0,8,8,0,0,15]),
bytearray([158,1,146,241,145,242,1,158,15,0,0,0,8,0,0,7])],
         [bytearray([243,230,152,18,101,235,239,31,15,15,15,7,2,0,10,15]),
bytearray([231,197,153,21,73,215,223,63,15,15,14,2,0,0,5,14])]]
EnemySet = EnemySets[0]

#test enemies
# BITMAP: width: 8, height: 12
bitmap7 = bytearray([62,1,146,241,145,242,1,62,7,0,0,8,8,0,0,15]),
bytearray([158,1,146,241,145,242,1,158,15,0,0,0,8,0,0,7])
```

```
# Mango
# BITMAP: width: 6, height: 8
CoinSpr = bytearray([135,115,88,70,126,129])


# TitleScreen
thumby.display.fill(0)
thumby.display.setFont("/lib/font8x8.bin", 8, 8, 0)
thumby.display.drawText("Mango", 0, 10, 1)
thumby.display.drawText("Run", 10, 20, 1)
thumby.display.blit(titleFace, 47, 10, 22, 14, -1, 0, 0)
thumby.display.setFont("/lib/font5x7.bin", 5, 7, 1)
thumby.display.update()

thumby.audio.playBlocking(2637, 125)
thumby.audio.playBlocking(2637, 125)
thumby.audio.playBlocking(20, 125)
thumby.audio.playBlocking(2637, 125)
thumby.audio.playBlocking(20, 125)
thumby.audio.playBlocking(2093, 125)
thumby.audio.playBlocking(2637, 125)
thumby.audio.playBlocking(20, 125)
thumby.audio.playBlocking(3136, 125)
thumby.audio.playBlocking(20, 125)
thumby.audio.playBlocking(20, 125)
thumby.audio.playBlocking(20, 125)
thumby.audio.playBlocking(1568, 125)

thumby.display.setFPS(60)

# Wait until the player is ready to play
while not thumby.buttonA.pressed():
    if time.ticks_ms() % 1000 < 500:
        thumby.display.drawFilledRectangle(0, 31, 72, 9, 0)
        thumby.display.drawText("Press A", 15, 32, 1)
    else:
        thumby.display.drawFilledRectangle(0, 31, 72, 9, 1)
        thumby.display.drawText("Press A", 15, 32, 0)
    thumby.display.update()
    pass

#Music Example
```

```
MusicNoteDict = { 0:40000,
            "C4":261,
            "D4":293,
            "E4":329,
            "F4":349,
            "G4":392,
            "A4":440,
            "B4":494,
            "C5":523,
            "D5":587,
            "E5":659,
            "F5":698,
            "G5":783,
            "A5":880}

#Overworld theme
SongList = ["E5","E5", 0  , 0  ,"C5","C5", 0  , 0  ,"A4", 0  ,"C5", 0  ,"C5","C5","C5", 0  ,
        "G4", 0  ,"C5", 0  ,"C5",  0 ,"G5", 0  ,"E5","E5", 0  , 0  ,"D5","D5", 0  , 0  ,
        "E5","E5", 0  , 0  ,"C5","C5", 0  , 0  ,"A4", 0  ,"C5", 0  ,"C5","C5","C5", 0  ,
        "G4", 0  ,"C5", 0  ,"F5","E5","D5", 0  ,"C5","C5","C5","C5", 0  ,"B4","C5","D5",]


NoteLengthMS = 200

NoteLengthUS = NoteLengthMS * 1000
SongLength = len(SongList) * NoteLengthUS

def PlayMusic(utimeTicksUS):
    CurSongBeat = int((utimeTicksUS % SongLength)/NoteLengthUS)
    CurNote = SongList[CurSongBeat]
    CurFreq = MusicNoteDict[CurNote]
    #print(CurFreq)
    thumby.audio.play(CurFreq, NoteLengthMS)
    return

BGMOffset = utime.ticks_us()
while GameRunning:
    t0 = utime.ticks_us() # Check the time

    #MusicStuff
    PlayMusic(t0 - BGMOffset)
```

```
    # Is the player on the ground and trying to jump?
    if JumpSoundTimer < 0:
        JumpSoundTimer = 0
    if (thumby.buttonA.pressed() == True or thumby.buttonB.pressed() == True) and YPos
== 0.0:
        # Jump!
        JumpSoundTimer = 200
        YVel = -2.5

    # Handle "dynamics"
    YPos += YVel
    YVel += Gravity
    Points += (XVel/2)
    JumpSoundTimer -= 15

    if JumpSoundTimer > 0:
        thumby.audio.set(500-JumpSoundTimer)
    #else:
    #thumby.audio.stop()

    # Accelerate the player just a little bit
    XVel += 0.000025

    # Make sure we haven't fallen below the ground
    if YPos > 0:
        YPos = 0.0
        YVel = 0.0

    # Did Mango hit an enemy?
    if EnemyPos < 7 and EnemyPos > -8 and YPos > -8:
        # Stop the game and give a prompt
        GameRunning = False

        thumby.display.fill(0)
        thumby.audio.stop()
        thumby.display.setFont("/lib/font8x8.bin", 8, 8, 0)
        thumby.display.drawText("GAME OVER", 0, 1, 1)
        thumby.display.drawText(str(int(Points)), 45, 13, 1)
        thumby.display.setFont("/lib/font5x7.bin", 5, 7, 1)
        thumby.display.drawText("Score:", 8, 13, 1)
        thumby.display.drawText("Play again?", 4, 24, 1)
        thumby.display.drawText("A:Y  B:N", 12, 32, 1)
```

```
    thumby.display.update()

    thumby.audio.playBlocking(250, 125)
    thumby.audio.playBlocking(250, 125)
    thumby.audio.playBlocking(20, 125)
    thumby.audio.playBlocking(200, 125)
    thumby.audio.playBlocking(200, 125)
    thumby.audio.playBlocking(20, 125)
    thumby.audio.playBlocking(120, 500)

    while not thumby.inputPressed():
        pass # Wait for the user to give us something

    while not GameRunning:
        if thumby.buttonA.pressed() == True == 1:
            # Restart the game
            XVel = 0.05
            YVel = 0
            YPos = 0
            Points = 0
            GameRunning = True
            EnemyPos = random.randint(72, 300)
            CloudPos = random.randint(60, 200)
            CoinPos = random.randint(60, 200)
            BGMOffset = utime.ticks_us()

        elif thumby.buttonB.pressed() == True:
            # Quit
            machine.reset()

# Did Mango collect a coin?
if (CoinPos > 0 and CoinPos < 8) and YPos < -4:
    thumby.audio.play(3136, 300)
    Points += 25
    CoinPos = random.randint(int(104+CloudPos), 300)

# Is the enemy out of view?
if EnemyPos < -24:
    Points += 10
    thumby.audio.play(440, 300)
    EnemyPos = random.randint(72, 450)
```

```
        randomEnemy = random.randint(0, len(EnemySets)-1)
        EnemySet = EnemySets[randomEnemy]

    # Is the cloud out of view?
    if CloudPos < -32:
        # "spawn" another one
        CloudPos = random.randint(72, 180)
        randomSky = random.randint(0, len(SkySets)-1)
        CloudSpr = SkySets[randomSky]

    # Is the coin out of view?
    if CoinPos < -12:
        CoinPos = random.randint(72, 180)

    # More dynamics
    EnemyPos -= XVel * 16
    CloudPos -= XVel * 2
    CoinPos -= XVel * 4

    # Draw game state
    thumby.display.fill(1)
    thumby.display.blit(CloudSpr, int(16 + CloudPos), 8, 16, 8, 1, 0, 0)
    thumby.display.drawFilledRectangle(int(6 + CoinPos), 8, 6, 8, 1) # Coin background so
it 'overwrites' clouds
    thumby.display.blit(CoinSpr, int(6 + CoinPos), 8, 6, 8, 1, 0, 0)

    if t0 % 250000 < 125000 or YPos != 0.0:
        # Player is in first frame of run animation
        thumby.display.blit(PlayerRunFrame1, 8, int(23 + YPos), 16, 16, 1, 0, 0)
    else:
        # Player is in second frame of run animation
        thumby.display.blit(PlayerRunFrame2, 8, int(24 + YPos), 16, 16, 1, 0, 0)

    if t0 % 250000 < 125000:
        thumby.display.blit(EnemySet[0], int(16 + EnemyPos), 28, 8, 12, 1, 0, 0)
    else:
        thumby.display.blit(EnemySet[1], int(16 + EnemyPos), 28, 8, 12, 1, 0, 0)

    thumby.display.drawText(fscore(int(Points)), 2, 1, 1) # Current points backdrop
    thumby.display.drawText(fscore(int(Points)), 1, 1, 0) # Current points
    thumby.display.update()
```

```
# Spin wheels until we've used up one frame's worth of time
while utime.ticks_us() - t0 < 1000000.0 / MaxFPS:
    pass
```