Kangaroo Mechanics

To start env use the RamStateDeltas Script in the scripts folder of JAXAtari repo with “python3 RAMStateDeltas.py -g Kangaroo”

Initial

- List of essential objects and how often they appear (max)
  - Player: 1
  - Child: 1,
  - Monkey: 4
  - FallingCoconut: 1,→ **Dropping down from the top**
  - ThrownCoconut: 3 → **Thrown at the player by the monkeys**
  - Fruit: 3 → **Collectable**
  - Bell: 1 → **Can be used to replenish fruits**
  - Ladder: 6 → **To reach higher platform**
  - **Platform**: 20

- Initial “Spawnpoint” of **Player**
  - X: 78, Y: 103
  - Width: 8, Height: 24
  - Crashed = False
  - Climbing = False

- Initial “Spawnpoint” of **Child**
  - X: 78, Y: 12
  - Width: 8, Height: 15

- Initial Platforms
  - Platform(16, 172, w=128)
  - Platform(16, 28, w=128)

- Current Level = RAM[36]
- TimeValue = RAM[59]
- Lives = RAM[45]

- Time (HUD)
  - x,y = 80,191
  - w,h = 15, 5
- Lives (HUD)
  - x,y = 16,183
- Score(HUD)
  - x,y = 129,183

Things to implement

- Basic Spawnpoints and Map Layout (hardcoded) → (should be fairly easy)
- Basic Player Movement and **interaction with map** (walking, crouching, jumping, **climbing**, **hitting**) + collision
- Movement of Child
- Movement of Monkeys
  - Throwing and movement of thrown coconuts
- Spawn and Movement of Falling Coconuts
- Behavior of Bell + Fruits

Player Movement

```
RAM[17] = X_Coord + 15
RAM[16] = Y_Coord * 8 + 4

RAM[18] = Counting which Sprite to use (also called orientation)
    -> 8 = Normal (right)
    -> 9 = Bouncing (also while normal walking NOT jumping) (right)
    -> 0 = Normal (left)
    -> 1 = Bouncing (left)
    -> When running -> iterating through 8 and 9 / 0 and 1
    -> 73 = Jump (right)
    -> 74 = Jump 2 (Smaller sprite) (right)
    -> 65 = Jump (left)
    -> 66 = Jump 2 (Smaller sprite) (left)
    -> 28 = Crouching (right)
    -> 20 = Crouching (left)

orientation = E if RAM[18] = 8,9,28,73,74 else orientation = W
climbing = True if RAM[18] = 39 or 47
crashed = True if RAM[18] = 1 or 128

Seems like RAM[19] is some kind of tick clock to use with the selection of different sprites (i guess)
```

- X_Coord (Walking) moved by using keys A and D
  - Changes by **1 every third frame/update**
- Y_Coord (Jumping/Crouching) moved by using keys W and S
  - Crouching:
    - Y_Coord +1 when starting to crouch and -1 when stopping
  - Jumping:
    - 8 Ticks after pressing W (read off the RAM[19] counter) Y_Coord -1 (-1 seems to be scaled by some value since the jump is not only one pixel)
    - After 16 Ticks the small sprite is displayed and the hitbox seems to be smaller
    - After 24 Ticks Y_Coord again -1
    - After 32 Ticks Y_Coord +1
    - After 40 Ticks Y_Coord +1 again → now back on ground

  → All 8 ticks something happens
- Player hight during jumping and crouching animation changes
  - If RAM[18] = 20, 28 → height = 16 (crouching)
  - If RAM[18] = 66, 74 → height = 15 (jump small)
  - If RAM[18] = 65, 73 → height = 23 (jump stretched)
  - Else → height = 24 (default)
- Player width = 8 (always)

- When hit, crash → fall down through all platforms

Child

```
If RAM[16] > 3
    Child_Y = 12
    Child_X = RAM[83] + 15

Else
    fruits = 0
    for i in range(3):
        if ram_state[42+i] & 128:
            fruits += 1
    if fruits == 0:
        fruits = 1
    if ram_state[68] == fruits:
        child.xy = ram_state[83] + 15, 12
```

- Seems to loop on the topmost platform

Monkey

```
for i in range(MAX_ESSENTIAL_OBJECTS["Monkey"]):
    if ram_state[11 - i] != 255 and ram_state[11 - i] != 127:
      x = ram_state[15 - i] + 16
    y = ram_state[11 - i] * 8 + 5
    if type(objects[2+i]) is NoObject:
      objects[2+i] = Monkey()
      objects[2+i].xy = x, y
  else:
    objects[2+i] = NoObject()
```

- Monkeys climb down on the left side (X is fixed and Y changes at first) when they reached player platform, they step towards player and occasionaly throw coconut either high or low so the player has to jump/crouch
- Sometimes also climb up
- When hit they despawn

Falling Coconut

```
if ram_state[33] != 255:
    x = ram_state[34] + 14
    y = (ram_state[33] - 22 * ram_state[36]) * 8 + 9
        if type(objects[6]) is NoObject:
            objects[6] = FallingCoconut()
        objects[6].xy = x, y
else:
    objects[6] = NoObject()
```

- When this one spawns it “bounces” on the topmost platform till it is above the player then drops straight down

Thrown Coconut

```
for i in range(MAX_ESSENTIAL_OBJECTS["ThrownCoconut"]):
    if ram_state[25 + i] != 255:
        x = ram_state[28 + i] + 15
        y = (ram_state[25 + i] * 8) + 1
        if type(objects[7+i]) is NoObject:
        objects[7+i] = ThrownCoconut()
        objects[7+i].xy = x, y
```

```
    else:
        objects[7+i] = NoObject()
```

Fruit

- When collected → Score increases (+100?)

Bell

```
lvl = ram_state[36]
if ram_state[41] == 128:
    objects[13] = NoObject()
elif lvl < 3:
    x = [93, 31, 130][lvl]
    y = 36
    if objects[13] is None:
    objects[13] = Bell()
    objects[13].xy = x, y
```

Platforms → Hardcoded

```
def manage_platforms(current_lvl_val, _):
    platforms = []

    # There is a total of 3 levels
    if current_lvl_val == 0:
        platforms = [
            Ladder(132, 132),
            Ladder(20, 85),
            Ladder(132, 37),
            NoObject(),
            NoObject(),
            NoObject(),
            Platform(16, 172, w=128), Platform(16, 28, w=128),
            Platform(16, 76, w=128),
            Platform(16, 124, w=128),
        ]
        platforms.extend([NoObject()]*16)

    elif current_lvl_val == 1:
        platforms = [
            Ladder(120, 132, h=4),
            Ladder(24, 116, h=4),
            Ladder(128, 36, h=4),
            NoObject(),
            NoObject(),
            NoObject(),
            Platform(16, 172, w=128), Platform(16, 28, w=128),
            Platform(16, 124, w=28), Platform(52, 124, w=92),
            Platform(16, 76, w=60), Platform(84, 76, w=60),
            Platform(28, 164, w=24), Platform(112, 84, w=24),
            Platform(120, 44, w=24), Platform(48, 156, w=32),
            Platform(76, 148, w=32), Platform(104, 140, w=32),
            Platform(16, 108, w=32), Platform(56, 100, w=20),
            Platform(84, 92, w=20), Platform(64, 60, w=20),
            Platform(92, 52, w=20), Platform(28, 68, w=28)
        ]
        platforms.extend([NoObject()]*2)
```

```
    else:  # current_lvl_val == 2
        platforms = [
            Ladder(20, 36, h=28),
            Ladder(20, 148, h=4),
            Ladder(36, 116, h=20),
            Ladder(104, 36, h=20),
            Ladder(120, 68, h=4),
            Ladder(132, 84, h=4), Platform(
                16, 172, w=128), Platform(16, 28, w=128),
            Platform(88, 140, w=16), Platform(
                64, 148, w=16), Platform(100, 116, w=16),
            Platform(48, 100, w=16), Platform(
                76, 52, w=16), Platform(80, 36, w=16),
            Platform(104, 132, w=20), Platform(
                84, 156, w=20), Platform(124, 124, w=20),
            Platform(52, 84, w=20), Platform(
                108, 164, w=36), Platform(16, 108, w=80),
            Platform(16, 92, w=28), Platform(
                76, 92, w=68), Platform(16, 140, w=32),
            Platform(96, 60, w=36), Platform(
                100, 76, w=44), Platform(60, 44, w=12)
        ]
```