(3) Interactive Mario Platformer

★ Things we'll learn

- Physics concepts — jumps, bounce, Gravity, speed
- keyboard controls
- Animatⁿ
- tweens (+ movⁿ bckgrnd) using arrays ds)
- Camera control (ie our screen goes whre the player moves)
- Groups logic
- Collision & Overlap

How to create land / series of tiles using one singele img:

Tiles →

```
function preload(){
    this.load.image("ground", "../Assets/..);
    this.load.image("sky", " _ );
}
```

bydefault we have img center as →

& to mk it

.setOrigin_(0,0)

```
function create(){
    W= game.config.width;
    H = game.config.height;

    let background = this.add.sprite(0,0,"sky");
    background.setOrigin (0,0);
    bagroud.displayWidth = W;
```

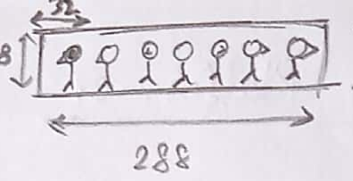
m#1 → mtlb 'W' tk strech kro

```
    let ground = this.add.~~sprite~~ tileSprite (0, H-128, W, 128, "ground");
    ground.setOrigin (0,0);
}
```

W → jaha tk lejana hai img ko ①

~~m~~ m#2 — 128 → height of the area jisme img fill krni hai.

How to load of player



} spritesheet

```
funct^n preload(){
    this.load.spritesheet("dude", "dude.png", {framewidth: 32,
                                               frameheight: 48});
```

each frame ki W & H kya hogi

Add physics concepts in the game :

```
let config = {
    :
    physics : {
        default: "arcade";
        arcade: {
            gravity: {
                y: 1000,
            },
            debug: true,
        }
    },
    :
}
```

y: 1000 → this val. of grav. works fine.

in 'arcade' debug: true → this shows the bounding boxes around all the objs on the screen.

(purple/pink lines reflect dynamic bodies & blue ones static ".)

green line shows the directn, larger the magnitude of velocity longer will be the green line.

Add^n physics to player :

```
fn create(){
    :

let player = this.physics.add.sprite(100,100, "dude", 4);
this.physics.add.existing(ground, true);
//ground.body.allowGravity = false;
//ground.body.imoravable = true;
this.physics.add.collider(ground, player);
```

with "true" makes the img static, that is imovable else all imgs by-default are dynamic.

existing → existn img pe physics lagana

allowGravity = false; → If "true", this img will also fall

imoravable = true; → irrespective of colisn this img won't move

collider(ground, player); → colisn detectn b/w these 2 imgs.

→ mtlb player ground se colid hoga then ground pe rest state pe a jaega.

dynamic
a group of objcts :

```
fn create(){
   :
   let fruits = this.physics.add.group ({
      key: "apple",  → jis img ka grp banana hai
      repeat : 8, → no. of imgs in grp
      setScale: {x:0.2, y:0.2}, → original img ka 20% ho ja
      setXY: {x: 10, y: 0, stepX:100},
                   ↳ for every repeatatn 'x' cord. will shift by 100.
   });
   :
}
```

M#1 to create attributes

Add Bounce effct on objcts

```
fn create(){
   :
   this.player.setBounce (0.2);
```

when set to 1, it'll mean that on every collisn there will be no energy loss ∴ it'll keep on bouncing. If x<1, x will mean there'll be loss of energy.

```
   fruits.children.iterate( function (f){
   for every objct of fruit iterate
        f.setBounce( Phaser.Math.floatBetween(0.4, 0.8));
   })
   :
}
```

for every objct we'll have dif value of bounce.

Add a static group of objs

```
fn create(){
   let platforms = this.physics.add.staticGroup();
   platforms.create(600,400,"ground").setScale(2, 0.5).refreshBody();
   — " ~~700~~,200  — " —
   — " ~~100~~, 200  — " —
   platforms.add(ground);
}
```

to 1se width 2 times

↳ ~~to~~ height by half.

↳ Now as we've reshaped the img, it's boundary have also changed, in order to set its boundary acc. to new scales, we use 'refreshBody()'.

↳ to add 'ground' in platforms container.

M#2 to add or create attributes

3

To check which key on keyboard is pressed:

(1) fn create(){

```
        :
    this.cursors = this.input.keyboard.createCursorKeys();
        :
}
```

(2) Now in update() fn we'll check which is presd:

```
fn update(){
   if (this.cursors.left.isDown){
      this.player.setVelocityX(-player_config.player_speed);
   }
   else if (this.cursor.right.isDown){
      this.player.setVelocityX(player_config.player_speed);
   }
   else {
      this.player.setVelocity(0);
   }

   if (this.cursors.up.isDown && this.player.body.touching.down){
      this.player.setVelocityY(player_config.player_jumpspeed);
   }
}
```

it mean when down arow key is prsd.

player img jo hai that is touch down ie is not in air. or img is touch" top of another img.

In main body we create an obj for player

```
let player_config = {
        player_speed : 150,
        player_jumpspeed : -700,
}
```

Addⁿ Animatⁿ

```
fn create(){
        :

   this.anims.create({
      key: "left",
      frames: this.anims.generateFrameNumbers("dude", {start:0,
                                                        end: 3}),
      frameRate: 10;
      repeat: -1,
   });
```

JSON objct

whn this anim. is called or trigrd: key

kaha se kaha tak frames chalni chahie: frames

frameRate → per second kitni frames dikhani

repeat → repeat for ∞ time

∴ like this we'll create for "right" & "center" facing.

for "right = {start:6, end:8}" & "centr = {st:4, e:4}".

- Now in update just called this animatⁿ, whn left key is
∴ prsd call "left" anim, similarly for right & center.

(ex)

```
fn update(){
      :
   if (←){
      :
      this.player.anims.play("left", true);
   }
 ∴ :
}
```

#OVERLAP: ie whn playr eats/overlaps fruit.

```
fn create(){
   :
   this.physics.add.overlap(this.player, fruits, eatFruit, null, this);
   :
}
```

eatFruit ↑ trigrs this fn() whn 1st para img/obj and 2nd para img/obj overlaps

eatFruit → colide callback fn() call

null ↓ for aditional checks [not needed!]

this ↓ context in which to run the callback fn.

```
Function eatfruit(player, fruit){
   fruit.disableBody(true, true);
}
```

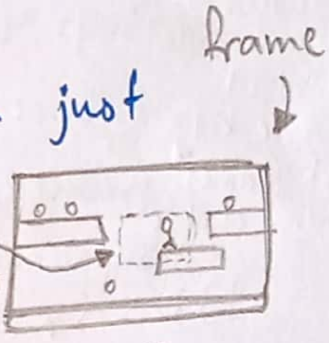
1st true → disable GameObject — deactivate game object

2nd true → hide GameObject (S) — only hides the object.

check that player doesn't go out of frame:

```
fn create(){
      :
  this.player.setCollideWorldBounds(true);
   :
}
```

Instead of showing the whole frame, we can just zoom our screen towards the player: (CAMERA)



```
fn create(){
      :
   this.cameras.main.setBounds(0,0,W,H);   → dimensions of camera screen.
 //this.physics.world.setBounds(0,0,W,H);

   this.cameras.main.startFollow(this.player, true, true); → to tell camera whom to focus on.
   this.cameras.main.setZoom(1.5);   → kitna zoom karna.
     :
}
```

Addn tweens for sunrays

```
fn create(){
     :
     let rays=[]
     for(let i=-10, i<=10, i++){
       let ray = this.add.sprite(W/2, H-100,"ray");
       ~~ray.displaywidth~~
      ray.displayHeight = 1.2 * H;
      ray.setOrigin(0.5,1);
      ray.alpha = 0.2;
      ray.angle = i*20;
      rays.push(ray);
     }
```

```
this.tweens.add({
   targets: rays,
   props: {
     angle:{
        value: "+=20",
     },
   },
   duration: 8000,
   repeat: -1,
});
```