

Prototype 3

The prototype three is constructed off of the second prototype. For prototype three our team has added movement for when the polar bear gets close to the animals causing them to run away. Additionally, we still had to fix some of the problems previously such as the animals moving in one straight line and not in random directions. We also added the hunger and stamina system to help immerse the player in the VR simulation. We also have added the second scene of the simulation of the present that the polar bear will be.

This was obtained by each group member going off and doing their parts and then coming together at the end of the week to add them together. This is done by using a USB and sticking it into the main laptop that we will be working on.

Code for hunger system:

```
using System;
using System.Collections;
using System.Collections.Generic;
using StarterAssets;
using UnityEngine;
using UnityEngine.Events;

public class survivlemanager : MonoBehaviour
{
    [Header("Hunger")]
    [SerializeField] private float _maxHunger = 100;
    [SerializeField] private float _hungerDepletionRate = 1;
    private float _currentHunger;

    public float HungerPercent => _currentHunger / _maxHunger;

    [Header("Stamina")]
    [SerializeField] private float _maxStamina = 100;
    [SerializeField] private float _staminaDepletionRate = 1;
    [SerializeField] private float _staminaRechargeRate = 1;
```

```

[SerializeField] private float _staminaRechargeDelay =  ;
private float _currentStamina;
private float _currentStaminaDelayCounter;
public float staminaPercent => _currentStamina / _maxStamina;

[Header("player References")]
[SerializeField] private StarterAssetsInputs _playerInput;

public static UnityAction OnPlayerDied;

private void Start()
{
    _currentHunger = _maxHunger;

    _currentStamina = _maxStamina;
}

private void Update()
{
    _currentHunger -= _hungerDepletionRate * Time.deltaTime;

    if(_currentHunger <=  )
    {
        OnPlayerDied?.Invoke();
        _currentHunger =  ;
    }

    if(_playerInput.sprint)
    {
        _currentStamina -= _staminaDepletionRate * Time.deltaTime;
        _currentStaminaDelayCounter =  ;
    }

    if(!_playerInput.sprint && _currentStamina < _maxStamina)
    {
        if ( _currentStaminaDelayCounter < _staminaRechargeDelay)
            _currentStaminaDelayCounter += Time.deltaTime;
    }
}

```

```

        if (_currentStaminaDelayCounter >= _staminaRechargeDelay)
        {
            _currentStamina += _staminaRechargeRate * Time.deltaTime;

            if (_currentStamina > _maxStamina) _currentStamina =
_maxStamina;
        }
    }

    public void ReplenishHungerThirst(float hungerAmount)
    {
        _currentHunger += hungerAmount;

        if (_currentHunger > _maxHunger) _currentHunger = _maxHunger;

    }
}

```

Code for moving:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[RequireComponent(typeof(CharacterController))]
public class FPSController : MonoBehaviour
{
    public Camera playerCamera;
    public float walkSpeed = ;
    public float runSpeed = ;
    public float jumpPower = ;
    public float gravity = ;

    public float lookSpeed = ;
    public float lookXLimit = ;
}

```

```

Vector3 moveDirection = Vector3.zero;
float rotationX = ;

public bool canMove = true;

CharacterController characterController;
void Start()
{
    characterController = GetComponent<CharacterController>();
    Cursor.lockState = CursorLockMode.Locked;
    Cursor.visible = false;
}

void Update()
{
    #region Handles Movment
    Vector3 forward =
transform.TransformDirection(Vector3.forward);
    Vector3 right = transform.TransformDirection(Vector3.right);

    // Press Left Shift to run
    bool isRunning = Input.GetKey(KeyCode.LeftShift);
    float curSpeedX = canMove ? (isRunning ? runSpeed : walkSpeed)
* Input.GetAxis("Vertical") : ;
    float curSpeedY = canMove ? (isRunning ? runSpeed : walkSpeed)
* Input.GetAxis("Horizontal") : ;
    float movementDirectionY = moveDirection.y;
    moveDirection = (forward * curSpeedX) + (right * curSpeedY);

    #endregion

    #region Handles Jumping
    if (Input.GetButton("Jump") && canMove &&
characterController.isGrounded)
    {
        moveDirection.y = jumpPower;
    }
    else
    {
        moveDirection.y = movementDirectionY;
    }
}

```

```

    }

    if (!characterController.isGrounded)
    {
        moveDirection.y -= gravity * Time.deltaTime;
    }

    #endregion

    #region Handles Rotation
    characterController.Move(moveDirection * Time.deltaTime);

    if (canMove)
    {
        rotationX += -Input.GetAxis("Mouse Y") * lookSpeed;
        rotationX = Mathf.Clamp(rotationX, -lookXLimit,
lookXLimit);
        playerCamera.transform.localRotation =
Quaternion.Euler(rotationX, , );
        transform.rotation *= Quaternion.Euler( ,
Input.GetAxis("Mouse X") * lookSpeed, );
    }

    #endregion
}
}

```

Prototyping Test Results:

Number	Test Description	Results
1	Hunger and stamina functionality	After writhing in the code we had to make the hunger drain a lot slower to make it fit in the time frame of the simulation. We also had to make the stamina rise slower so that the player isn't always sprinting. Other than that the results came back positive
2	Movement of ai	When adding in the movement of the AI at first the animals were running in straight lines. But after a few modifications, they can now go in different directions without hitting objects. Animations are attached to the animals to make it more immersive.
3	Adding in the	When adding in a

	second scene	second scene it wasn't too bad we just had to add in less ice as well as make it accurate to today's standard.
--	--------------	--

Feedback:

Feedback from the game was positive such as liking how we incorporated the hunger and stamina mechanics for the polar bear. We were also asked how we could chase the scenes from scene one to scene two. We were thinking about using a clock to show how the time changes but right now in our state of work we might just have to add in a text saying so and so years later.