**TEMASEK POLYTECHNIC**

**SCHOOL OF INFORMATICS & IT**

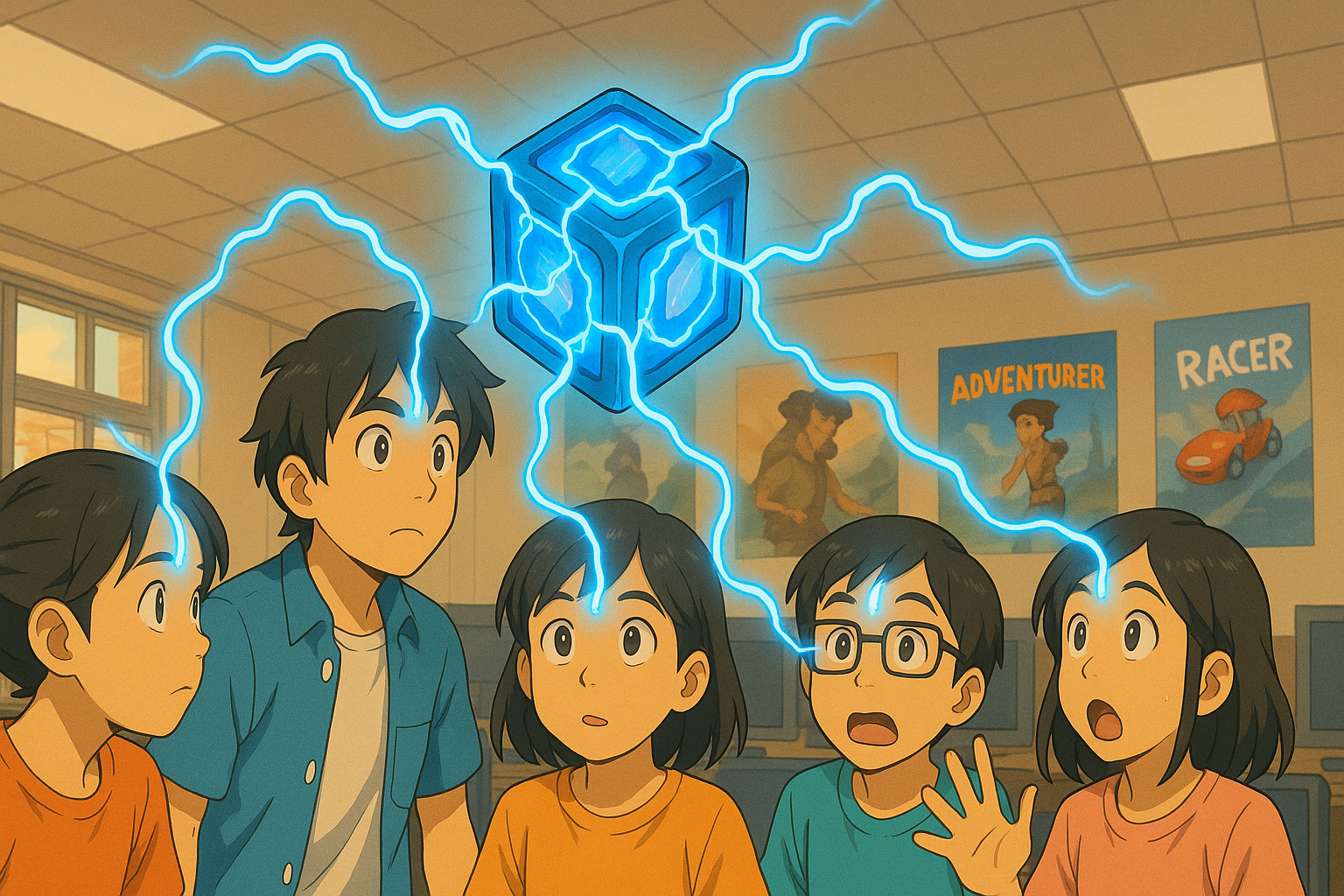
**DIPLOMA IN IMMERSIVE MEDIA & GAME DEVELOPMENT**

**AY2025/2026 APRIL SEMESTER**

**GADV (CGE2C25)**

**Unity Physics Worksheet**

To see the additional comments and resources, make sure you select **All Markup** in the **Review/Tracking** pane



**QUESTION 1 (basics of Unity physics)**

1. What is the core responsibility of a RigidBody component?

The Rigidbody component allows the GameObject that it has been attached to, be affected by physics such as gravity, collision, force application

1. Why is the Pong game paddle mentioned as an example of an object whose rigidbody’s isKinematic property would be set to true? Hint: what controls the paddle movement?

The player controls the paddle movement, therefore the isKinematic property is set to true as the rigidbody of the paddle should be affected by the player and not, external forces in the game scene that is being applied affect the paddle too.

1. Explain what the rigidbody **constraints** are for. Give ONE example of when this might be useful.

Rigidbody constraints restricts the Gameobject from being affected by physics in certain way and not move in the way that is not wanted by the creator. One example would be pushing a box, rigidbody constraints can be used to prevent a box in the game from being flipped or lifted if it is lighter than the player, which is useful in puzzle games.

1. Watch [this video](https://www.youtube.com/watch?v=ixM2W2tPn6c).
2. What is one major problem with using the Translate function to move an object?

It does not regard any physics and colliders in the game. If you want to restrict the object to not pass through walls or fall down, translate does not regard the physics and moves the object through that position as it only focuses on the position of the object and not what other objects the moved object is currently interacted with.

1. Why is the physics-based code put in the FixedUpdate function and not in Update?

FixedUpdate happens every fixed interval. This keeps the physics consistent and for it all to be applied at once to prevent inaccurate forces to be applied as update may happen slower or faster than fixedUpdate and make the

1. What are the THREE different methods covered in the video to move an object using physics?
2. rb.AddForce()
3. rb.Velocity()
4. rb.MoveObject()

Explain how each method works.

rb.AddForce() pushes the object in the direction with the force of the speed. This makes the object initially slow, and proceeds to build up speed. The object also needs to slow down before coming to a stop. This could be used for objects with wheels or an object on a slippery surface like oil or ice.

rb.Velocity() overrides all physics elements including gravity. There is no acceleration using moving the object using this method. It is either moving, or stationery.

rb.MoveObject() is just like translations, which moves the object around the scene, but forces of gravity and physics work on it, which is good for a player model moving around the world.

**QUESTION 2 (Colliders)**

1. What is the core responsibility of a collider component?

A collider component defines a shape in space, it is basically an box that tells the game where an object is, if something is touching the box, the system detects it as something that has collided with the object and will push out logic for when the object has with another object

1. Look at the question in [this forum post](https://forums.oculusvr.com/developer/discussion/59641/problems-with-fast-moving-object-collision-in-unity-table-tennis-racket-vs-ball). What is the solution to the problem? Do some research to explain why this is the solution. Draw a diagram to illustrate your answer.

The solution is to change the collision type to dynamic. The physics are checked every few frames and not every frame to save memory and make the game optimized. Dynamic checks every frame so if something is moving faster than inbetween the checks, it will still count it and allow the colliders to detect that something has collided.

1. What is the difference between collision *detection* and collision *resolution*?   
   Collision detection checks if something has overlaps the collision box

Collision resolution tells the system what to do when something has overlapped the collision box.

**QUESTION 3 (Joints)**

1. What type of joint is used in the video? According to Unity’s documentation, what does this joint do? Give an example of how the joint could be used in a game.

Hinge Joint: connects the two object with the joints being a pivot, used to make the doors in games

1. What other types of joints are there? Briefly explain what FOUR of these joints does (do NOT include the type of joint you answered for part a above), and give an example of how each one might be used in a game.

Fixed Joint: Attach the two object with rigidbody together making them dependant of each other its good to connect two objects together without parenting. A moving platform with objects on top of it to allow the objects to not move about while the platform is moving back and forth

Spring Joint: Creates an effect very similar to a rubber band or a slingshot. Helps to simulate swings in game (like ropes and vines)

Character Joint: Character joints give you a lot of possibilities for constraining motion like with a universal joint. It is used to create ragdoll effects in games

Configurable Joint: There are two primary functions that the Configurable Joint can perform: movement/rotation restriction and movement/rotation acceleration. It is used for unnatural objects that require specific rotation to not be seem unnatural which the other joints do not offer.

**QUESTION 4 (Physic Material)**

1. What is the difference between dynamic friction and static friction?

Dynamic friction is how much friction exists while the object is moving

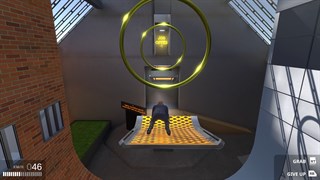
Static friction is how much force is required to move the object from its static position

1. Find a game that you have seen online or played yourself where physics is a core game mechanic.

You must:

* give a brief description of the game
* briefly explain how physics is used
* explain why physics is important
* include a relevant screenshot of the game

“Get to work” The game is a ragegame focused on sliding to the top of the platforms to ultimately become CEO. Physics plays a big part of the game as it allows the character to roll around the ramps, build up speed and lunge themselves into the air to reach another platform. There are different types of platforms which trigger different physics effects including gripping onto the floor to increase their own friction to stop acceleration. Physics is a main part of the game as the player must utilize physics to their advantage to avoid slipping off the platform and falling

**QUESTION 5 (CharacterController)**

1. According to the video, the built-in CharacterController component does not interact with physics. So, a CharacterController does not react to gravity, and when a CharacterController collides with another object with a rigidbody it does not have any force added to it (like when being hit by a cannonball throws an object across the scene).

What gameplay-related reason(s) can you think of to explain why the CharacterController component was designed this way?   
It was meant to stop the player from being thrown across when something interacts with it, for example in Mario when you get hit. You flash but you stay on the same spot while you take damage. Allowing physics would make your character fly back to the start.

1. To make a character controlled by a CharacterController experience gravity while moving, two options are given in the video. What are these? Dynamic Mode and CharacterController.SimpleMove
2. Create a Unity scene with a capsule game object with a CharacterController. The script below will make the character jump when the space bar is pressed.

Fill in the missing parts of the code, then copy the whole Character class code here. Make sure it is formatted properly!

**public class CharacterMovement : MonoBehaviour**

**{**

**CharacterController characterController;**

**public float speed = 6.0f;**

**public float jumpSpeed = 8.0f;**

**public float gravity = 20.0f;**

**private Vector3 moveDirection = Vector3.zero;**

**void Start()**

**{**

**characterController = GetComponent<CharacterController>();**

**}**

**void Update()**

**{**

**if (characterController.isGrounded)**

**{**

**moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0.0f, Input.GetAxis("Vertical"));**

**moveDirection \*= speed;**

**if (Input.GetButton("Jump"))**

**{**

**moveDirection.y = jumpSpeed;**

**}**

**}**

**moveDirection.y -= gravity \* Time.deltaTime;**

**characterController.Move(moveDirection \* Time.deltaTime);**

**}**

**}**

**QUESTION 6 (Constant Force component)**

1. For the Constant Force component, what is the difference between the Force and Relative Force properties, and between the Torque and Relative Torque properties?

Force is based on world space but relative force is affected by the local space (the direction the object is facing)

Torque and Relative Torque is the game, applying a rotation force in world space and local space.

1. When the cube is in the air with a Force X of 1, it moves, but when it falls to the plane, it doesn’t move.

Why not?

When the cube is in the air, there is no friction but when its touching the plane, the gravity creates friction and the cube does not have enough force to overcome the static friction and stays in place.

1. An object has a constant force applied along its local positive Z axis. What happens to the object’s speed over time?

Explain your answer.

The object would gain speed over time as more force is added in, increasing its acceleration. The formula for movement is movement \* force. As the force keeps increasing, the movement will increase leading the cube to go faster and faster

**QUESTION 7 (Forces)**

1. Paste your **Player** class code here. Make sure your code is readable, properly formatted, and commented.

using UnityEngine;

using System.Collections;

public class ExplosiveCharacterController : MonoBehaviour

{

CharacterController controller;

// Variables for moving the player

public float speed = 6.0f;

public float jumpSpeed = 8.0f;

public float gravity = 20.0f;

// Explosion variables

public float radius = 100.0f;

public float power = 10000.0f;

private Vector3 moveDirection = Vector3.zero;

void Start()

{

controller = GetComponent<CharacterController>();

controller.detectCollisions = false; // Only set to false if you want to ignore collisions!

}

void CheckExplosion()

{

if (Input.GetKeyDown(KeyCode.E))

{

Collider[] colliders = Physics.OverlapSphere(transform.position, radius);

Debug.Log("Found " + colliders.Length + " colliders in explosion radius.");

foreach (Collider hit in colliders)

{

Rigidbody rb = hit.GetComponent<Rigidbody>();

if (rb != null)

{

rb.AddExplosionForce(power, transform.position, radius, 3.0f);

}

}

}

}

void FixedUpdate()

{

CheckExplosion();

if (controller.isGrounded)

{

moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0.0f, Input.GetAxis("Vertical"));

moveDirection \*= speed;

if (Input.GetButton("Jump"))

{

moveDirection.y = jumpSpeed;

}

}

moveDirection.y -= gravity \* Time.deltaTime;

controller.Move(moveDirection \* Time.deltaTime);

}

}

1. Paste your **Kick** function code here. Make sure your code is readable, properly formatted, and commented.

void Kick()

{

if (Input.GetKeyDown(KeyCode.K))

{

// Define the center of the kick area a short distance in front of the player

Vector3 kickCenter = transform.position + transform.forward \* 2.0f; // 2.0f is the kick reach, adjust as needed

float kickRadius = 2.0f; // Area of effect for the kick, adjust as needed

Collider[] colliders = Physics.OverlapSphere(kickCenter, kickRadius);

foreach (Collider hit in colliders)

{

Rigidbody rb = hit.GetComponent<Rigidbody>();

if (rb != null)

{

// Apply force in the player's forward direction

rb.AddForce(transform.forward \* kickStrength, ForceMode.Impulse);

}

}

}

}

1. What does the upwardsModifier argument of AddExplosiveForce do?

it adds an extra upward force to the objects.



**QUESTION 8 (Forces cont.)**

1. What are the FOUR different values for ForceMode? Explain what each value does.

The force, the direction, the radius and the extra add upwards force.

Force: how strong the force being given is

Direction: the direction of the force

Radius: The area around being affected  
Upwards Modifier: Extra upward force above the force given to make things fly.

1. Paste your code for firing the sphere here.

using UnityEngine;

public class Impulse : MonoBehaviour

{

public float force = 10f;

private Rigidbody rb;

void Start()

{

rb = GetComponent<Rigidbody>();

}

void Update()

{

if (rb == null) return;

if (Input.GetKeyDown(KeyCode.UpArrow))

{

rb.AddForce(transform.forward \* force, ForceMode.Impulse);

}

else if (Input.GetKeyDown(KeyCode.DownArrow))

{

rb.AddForce(-transform.forward \* force, ForceMode.Impulse);

}

else if (Input.GetKeyDown(KeyCode.LeftArrow))

{

rb.AddForce(-transform.right \* force, ForceMode.Impulse);

}

else if (Input.GetKeyDown(KeyCode.RightArrow))

{

rb.AddForce(transform.right \* force, ForceMode.Impulse);

}

else if (Input.GetKeyDown(KeyCode.Space))

{

rb.AddForce(transform.up \* force, ForceMode.Impulse);

}

}

}

1. Paste your code for rotating the beam here.

using UnityEngine;

public class Impulse : MonoBehaviour

{

public float torque = 10f;

private Rigidbody rb;

void Start()

{

rb = GetComponent<Rigidbody>();

}

void Update()

{

if (rb == null) return;

else if (Input.GetKeyDown(KeyCode.Z))

{

// Clockwise rotation (negative Y torque)

rb.AddTorque(Vector3.down \* torque, ForceMode.Impulse);

}

else if (Input.GetKeyDown(KeyCode.X))

{

// Anti-clockwise rotation (positive Y torque)

rb.AddTorque(Vector3.up \* torque, ForceMode.Impulse);

}

}

}

**QUESTION 9 (Collision event-handling)**

1. Paste your code for **CosmicCube.cs** and **Orb.cs** here. Make sure your code is readable, properly formatted, and commented.

using UnityEngine;

public class CosmicCube : MonoBehaviour

{

public Material defaultMaterial;

public Material collisionMaterial;

private Renderer rend;

void Start()

{

rend = GetComponent<Renderer>();

if (rend != null && defaultMaterial != null)

{

rend.material = defaultMaterial; //Changes whatever material it is to default material

}

}

void OnCollisionEnter(Collision collision)

{

if (collision.rigidbody != null && collisionMaterial != null && rend != null)

{

rend.material = collisionMaterial; //When collided, it changes to collision material (in this case its green)

}

}

void OnCollisionExit(Collision collision)

{

if (collision.rigidbody != null && defaultMaterial != null && rend != null)

{

rend.material = defaultMaterial; //changes back the renderer material to the default (red) when no longer colliding.

}

}

using UnityEngine;

public class Orb : MonoBehaviour

{

public float impulse = 10f; // Dummy Value

private Rigidbody rb;

void Start()

{

rb = GetComponent<Rigidbody>(); //get rigidbody to do rigid body stuff

}

// Update is called once per frame

void Update()

{

if (rb != null && Input.GetKeyDown(KeyCode.Space))

{

rb.AddForce(transform.forward \* impulse, ForceMode.Impulse); //impulseforce code

}

}

}

**QUESTION 10 (Raycasting)**

1. Paste your completed code for the CheckLineOfSight() function here.

using UnityEngine;

public class \_PlayerController : MonoBehaviour

{

CharacterController characterController;

public float speed = 6.0f;

public float jumpSpeed = 8.0f;

public float gravity = 20.0f;

private Vector3 moveDirection = Vector3.zero;

public Material greenMaterial;

public Material redMaterial;

void Start()

{

characterController = GetComponent<CharacterController>();

}

void Update()

{

if (characterController.isGrounded)

{

moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0.0f, Input.GetAxis("Vertical"));

moveDirection \*= speed;

if (Input.GetButton("Jump"))

{

moveDirection.y = jumpSpeed;

}

}

moveDirection.y -= gravity \* Time.deltaTime;

characterController.Move(moveDirection \* Time.deltaTime);

CheckLineOfSight();

}

void CheckLineOfSight()

{

GameObject[] enemies = GameObject.FindGameObjectsWithTag("Enemy");

RaycastHit hit;

foreach (GameObject enemy in enemies)

{

Renderer enemyRenderer = enemy.GetComponent<Renderer>();

if (enemyRenderer != null)

{

enemyRenderer.material = redMaterial;

}

}

foreach (GameObject enemy in enemies)

{

Vector3 direction = enemy.transform.position - transform.position;

Debug.DrawRay(transform.position, direction, Color.red, 0.1f);

Renderer enemyRenderer = enemy.GetComponent<Renderer>();

if (enemyRenderer == null) continue;

if (Physics.Raycast(transform.position, direction.normalized, out hit, 30f))

{

if (hit.collider.gameObject == enemy)

{

Debug.DrawRay(transform.position, direction, Color.green, 0.1f);

enemyRenderer.material = greenMaterial;

}

}

}

}

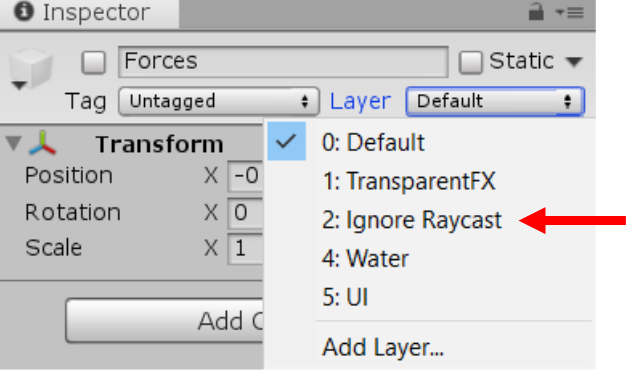
}

1. When checking for a hit using a ray, the code below is used:

if (Physics.Raycast(transform.position, vec, out hit, vec.magnitude))

Why is **vec.magnitude** used for the length of the ray? Because magnitude has direction and distance. Not only does the ray need to have length but it also needs to have a direction to move forward.

1. The layers dropdown in the Inspector has an entry for **Ignore Raycast**.



What does this mean? Why do you think this is important? Give an example.

It means the object will not be calculated in raycasting. For example if its blocking a box, the player will see the box even though the object is infront and blocking the line of sight. This can be good for glass or decoration that you want to be able to see through.