**Transform**

**\_**կա **աշխարհիկ** և **լոկալ** կոորդինատներ․ **Աշխարհիկը** տալիսա հենց էկրանի կոորդինատներն են սկսած ձախ ներքևից, իսկ **լոկալը**՝ կոորդինատները **parent** –ի նկատմամբ

***Թե ինչպես ստանալ թաչի աշխարհիկ կոորդինատները***

void Update()

{

Touch touch = Input.GetTouch( 0 );

if ( Input.GetKeyDown ( KeyCode.Mouse0 ) ) // es animasta qani vor inqy obyektin tanelua tuchi positioni vra voch te clicki

{

MyGameObject.transform.position = Camera.main.ScreenToWorldPoint (new Vector3 ( touch.position.x, touch.position.y ,Camera.main.nearClipPlane ) );

}

}

\_Ոնց հասկացա **աշխարհիկ կոորդինատները** տրվում են **Camera.main.**Screen… կլասսով, որը ունի տարբեր տիպի աշխարհիկ կոորդինատներ, այդ թվում պիքսելային և նորմալիզացված ( 1-ով , որը շատ հարմարա)

**Camera.main.ScreenToWorldPoint(myVector)** // տալիսա transform pos-ը աշխարհիկով (unitվմիավորներով)

**Camera.main.WorldToScreenPoint(myVector)** // տալիսա պիքսելային կոորդինատները

\_իրականում եթե կոդում գրենք **input.pousePossition** ինքը հենց տալիսա աշխարհիկ կոորդինատներ, այսինքն էկրանի պիքսելներով, սակայն օրինակ **gameObject.transform.position** \_ը տալիսա յունիթիի միավորներով, ուստի **պիքսելներով սարքելու համար պետքա հետևյալը գրել**․

***Սա ստիպումա գնդակին տռչել քլիքի ուղղությամբ կախված արագությունից***

[SerializeField]

private GameObject ball;

private Rigidbody2D rb;

bool isJump = false;

void Start()

{

rb = ball.GetComponent<Rigidbody2D>();

}

void Update()

{

isJump = Input.GetMouseButtonDown(0);

}

private void FixedUpdate()

{

if (isJump)

{

rb.velocity = (Input.mousePosition - Camera.main.WorldToScreenPoint(ball.transform.position))/50;

Debug.Log(rb.velocity.x + " " + rb.velocity.y);

isJump = false;

}

}

**GameObject \_**ին ուղղությունով արագություն (vector) տալու համար օգտագործում են **Rigidbody** \_ի **velocity** կամպանենտը սքրիփթից, **ինչպես ցույց է տրված վերոհիշյալ օրինակում**

**UI**

**\_**Կամայական ui էլեմենտի հնարավորա տալ **button էֆֆեկտ**, այսինքն **որպեսզի վիզուալ տեսնենք սեղմումը և նշենք սեղման ժամանակ կանչվող ֆունկցիան** ․ դրա համար ավելացնում ենք button կամպանենտ․ էտ **button component** \_ը ունի հնարավորություն միանգամից օր․ click \_ի ֆունկցիան տալ (դաժե պարամետրերով ֆունկցիաներ ․ պարամետրն էլ նույնպես կարանք հենց տեղից նշել) Դրա համար պետքա նշել նաև այն game object \_ը, որի script \_ը պիտի օգտագործել   
**\_**UI \_ի էլեմենտը ծիպը կոնտեյներ սարքելու համար պետքա **canvas group** component ավելացնել, որի միջոցով կարանք կարգավորել նրա թափանցիկությունը (նաև **DoTween**\_ի միջոցով), սակայն, եթե դաժե դարձնենք 0, ինքը խաղի ժամանակ չի երևա բայց եթե սեղմենք կսեղմվի (video\_7)

//տալիսա իրան ուղղանկյունաձև տված ռամկա-կենտրոնից հեռավորությունները

**selectingImage.GetComponent<RectTransform>().anchorMax(Min) = new Vector2(0.99f, 0.23f);**

**TOOLS**

pivot-ը կա ամեն Game object-ի համար և ցույցա տալիս վերջինիս կենտրոնը

վերևի ղեկավարման դաշտի **center ( pivot )** —ով ասվումա, թե որպես բռնիչ ընրենք իրական կենտրոնը, թե pivot\_ով նշված կետը

**sprite editor**

\_նոր մտած պատուհանում կանաչ ուղղանկյունաձև գծերով նշում ենք, թե սքեյլի ժամանակ ոնց պիտի իրեն պահի սպրայթը․ ուղղանկյան մեջ նշված մասը մնալուա հաստատւն իսկ իրա անկյունների 4 կտորները սեղմվելու են կամ ձգվեն․ որպեսզի այդ սպրայթի game object\_ը իրեն վերոհիշյալ ձևով պահի, անհրաժեշտա **inspector** \_ի **image type** \_ն ընտրել **slice**

\_վերևի ձախ անկյունից եթե նշենք **Custom Outline** , ապա պելիգոններով կարանք նշենք, թե էտ սպրայթը իրեն ոնց կպահի **polygon collider** \_ի ժամանակ

**Collider**

**polygon collider** \_ով տալիս ենք game object-ի իրական ձևը, սակայն դրա rendering-ը շատ դանդաղա կատարվում

**Prefab**

**Prefab \_**ը օգտագործվումա եթե նույն օբյեկտից պետքա խաղում մի քանի անգամ ստեծել կամ ջնջել ինչպես **edit time** այնպես էլ **run time :** Ստեղծելու համար պետքա մեր GameObject \_ ը Hiearchy\_ից քաշել քցել Asset\_ների մեջ, հետո ճիշտ կլինի Hiearchy –ից ջնջել, script –ի մեջ [SerializeField]

GameObject ենք սարքում ու իրա մեջ քաշում քցում ենք էտ **Prefab \_**ը։

**Սա կստեղծի *ball* պրեֆաբ-օբյեկտից մկնիկը սեղմած պահած կետում — այսինքն անհնդհատ կստեղծի**

[SerializeField]

private GameObject ball;

void Update()

{

if (Input.GetMouseButton(0))

{

Vector3 vv = new Vector3( Camera.main.ScreenToWorldPoint (Input.mousePosition).x, Camera.main.ScreenToWorldPoint(Input.mousePosition).y) ;

Instantiate(ball, vv, Quaternion.identity);

}

}

**Անիմացիա**

**Unity**\_ում կա անիմացիայի մի լավ դվիժոկ, որը կոչվումա **DoTween**

**using DG.Tweening;**

//տեղափոխումա 1․ կոորդինատը 2․ վայրկյանի ընթացքում

gObject.transform.DOMove( new Vector3( selectedMenuPanel.transform.position.x, Screen.height / 10 \* 9 , 0 ), 2 );

//փոխումա գույնը 1․-ի 2․վայրկյանի ընթացքում

gObject.GetComponent<Image>().DOBlendableColor(Color.white, 2);

// **canvas group**  կամպանենտ ունեցող UI օբյեկտի թափանցիկությունը

menuPanels[4].DOFade(1f, 0.5f);

կյկյլ

լլլյլյլյլլլլլլլլյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյյ

**Rigidbody**

**Rigidbody.velocity** արագություննա տալիս ու ստանում

**Rigidbody.AddForce** պտտմանուժա տալիս

***// կվերադարձնի քլիքին տրված ֆունկցիայում քլիք արած գեյմՕբյեկտը***

**EventSystem.current.currentSelectedGameObject.name**

***// Object \_ը կոդից clone անելու համար օգտագործվումա* Instantiate *ֆ-ն***

GameObject selectingImageClone = Instantiate(selectingImage, selectingImage.transform.parent);

// երկրորդ պարամետրով տրվում է կլոնավորված օբյեկտի **parent**\_ը, քանի որ այն լռությամբ չի տրվում, եթե մի պարամետրանոց // տարբերակն օգտագործենք

**Trail Renderer** հետագիծ ցույց տալու համարա

**Line Renderer** նմանատիպ բանա, օրինակ պարան սարքելու համար

DontDestroyOnLoad(this); ------------ pahpanuma obyekty bolor scenanerum