# Durante la realización de este proyecto se usaron varias herramientas de Unity para la implementación del juego. Entre lo utilizado podemos encontrar Coliders los cuales fueron la base para realizar la mayoría de las partes en el proyecto. Entre los códigos más interesantes en el uso de estos fue en los que los Coliders de las zonas de unidad se inhabilitan mientras se muestre el alguna caja de información(Box Info) o se tenga algún objeto en la mano seleccionado mientras estos se desactivan para poder tener control sobre los Coliders y se puedan ver luego en la Card Description al moverlos por encima( La Card Description fue realizada en otra parte del código).

public static void CheckColliderZone(GameObject gameObject, Player Owner)

{

*try*

{

*if* (Gwent.BoxInfo.GetComponent<BoxInfo>().Active) gameObject.GetComponent<Collider2D>().enabled = false;

*else if* (gameObject.GetComponentsInChildren<DisplayCard>() != null && gameObject.GetComponentsInChildren<DisplayCard>().Count() > 0)

{

*if* (!Gwent.InitialDraw && Owner.SelectedCards != null && Owner.SelectedCards.Count > 0 && Owner.Hand.GetComponentsInChildren<DisplayCard>().Contains(Owner.SelectedCards.Last(

).GetComponent<DisplayCard>()))

{

gameObject.GetComponent<Collider2D>().enabled = true;

}

*else*

{

gameObject.GetComponent<Collider2D>().enabled = false;

}

}

*else*

gameObject.GetComponent<Collider2D>().enabled = true;

}

*catch*

{

}

Otra parte del proyecto que puede ser de interés es la antes mencionada Card Description en el que tenemos un objeto Image que se encarga de cargar el Sprite de la carta a la cual el mouse está apuntando, un text info que permite observar el efecto que cada carta presenta y otro text el cual nos da el ataque de la carta en caso de ser de tipo unidad( el ataque de cartas especiales y del señuelo son equivalentes a 0 y no se muestran en el text).

private void OnMouseEnter()

{

*if* (card.Owner.IsMyTurn ||

!card.Owner.Hand.GetComponentsInChildren<DisplayCard>().Contains(*this*))

{

cardView.GetComponent<Image>().sprite = sprite;

*if* (card.Atk.ToString() == "0")

{

}

cardView.GetComponent<CardView>().Atk.GetComponent<Text>().text =

"";

}

*else*

{

cardView.GetComponent<CardView>().Atk.GetComponent<Text>().text = card.Atk.ToString();

}

cardView.GetComponent<CardView>().Effect.GetComponent<Text>().text = card.Description;

# Para establecer diferencias entre ambos turnos y evitar que ambos players vean sus manos se decidió relaizar algo similar a un efecto espejo en el que mientras sea el turno del jugador los sprites de las cartas del oponente serán un sprite genérico que fue establecido como la parte trasera de la carta.

private void OnMouseDown()

{

*if* (card.Owner.IsMyTurn)

{

*if* (card.Owner.SelectedCards.Contains(gameObject))

{

card.Owner.SelectedCards.Remove(gameObject);

}

*else*

{

card.Owner.SelectedCards.Add(gameObject);

}

}

*if*

(!Gwent.Player1.Hand.GetComponentsInChildren<DisplayCard>().Contains(*this*) &&

!Gwent.Player2.Hand.GetComponentsInChildren<DisplayCard>().Contains(*this*))

{

*if* (card.Owner.SelectedCardsBoard.Contains(gameObject))

{

card.Owner.SelectedCardsBoard.Remove(gameObject); card.Owner.SelectedCardsBoard.Remove(gameObject);

}

*else*

{

card.Owner.SelectedCardsBoard.Add(gameObject); card.Owner.SelectedCardsBoard.Add(gameObject);

}

}

}

# Dentro del código hay una parte que está dedicada al control de los efectos, los cuales permiten pasar de turno(reglas del juego en el readme), mediante este hacemos un switch que permite verificar los efectos y ver cuales mantienen los turnos hasta la ejecución de su efecto y así, una vez ejecutado sea el turno del otro jugador.

public IEnumerator ActivateEffect(DisplayCard displayCard)

{

Player Owner = displayCard.card.Owner;

BoxInfo boxInfo = BoxInfo.GetComponent<BoxInfo>(); bool EndTurn = true;

*switch* (displayCard.card.Effect)

{

*case* Effects.None:

*break*;

*case* Effects.SetCardSpecialInOwnRow:

*yield return* SetCardSpecialInOwnRowEffect(Owner); EndTurn = false;

*break*;

*case* Effects.SetWeather:

*yield return* StartCoroutine(SetWeatherEffect(Owner)); EndTurn = false;

*break*;

*case* Effects.EliminateBiggestAtkCard:

*yield return* DestroyMonsterWithHigherOrLessAtk(true);

*break*;

*case* Effects.EliminateLeastAtkCard:

*yield return* DestroyMonsterWithHigherOrLessAtk(false);

*break*;

*case* Effects.Draw: DrawEffect(Owner); *break*;

*case* Effects.ClearRowLessUnit: ClearRowLessUnitEffect(Owner); *break*;

*case* Effects.BalanceAtkField:

*yield return* StartCoroutine(BalanceEffect(boxInfo));

*break*;

*case* Effects.Decoy:

*yield return* StartCoroutine(DecoyEffect(Owner, DecoyCard)); EndTurn = false;

*break*;

*case* Effects.Decoy2:

*yield return* StartCoroutine(DecoyEffect(Owner, DecoyCard2)); EndTurn = false;

*break*;

*case* Effects.WeatherOff: WeatherOffEffect(); *break*;

*case* Effects.DestroyCard:

*yield return* StartCoroutine(DestroyEffect(Owner)); EndTurn = false;

*break*;

*case* Effects.Invocation: InvocationEffect(Owner); EndTurn = false;

*break*;

*case* Effects.PowerUpSanctuary: PowerUpSanctuaryEffect(Owner, 2); *break*;

*case* Effects.PowerUpEvocation: PowerUpSanctuaryEffect(Owner, 4); *break*;

*case* Effects.WeatherStonehedge:

*yield return* StartCoroutine(ZoneSelectedInmune(boxInfo));

*break*;

*case* Effects.WeatherSmoke:

*yield return* StartCoroutine(ZoneSelectedNegateEffect(boxInfo));

*break*;

}

*if* (EndTurn)

SwitchTurn();

}

# Entre los efetos que más interesantes de implementar se vieron en el trabajo están los de los climas creados, los cuales permiten negar los efectos de las unidades de una zona seleccionada, y de impedir que las unidades de una zona seleccionada sean afectadas por efectos de cartas. Para esto se emplearon corutinas que esperan por una box info que permite seleccionar las zonas en las que se activarán los efectos mencionados, luego se le agrega a la zona el negar el efeto de las unidades o volverlas inmunes.

private IEnumerator ZoneSelectedNegateEffect(BoxInfo boxInfo)

{

*yield return* StartCoroutine(SelectZoneBox(boxInfo));

zonesSelectedNegate.Add(gwent.GetComponent<Gwent>().zonesSelect[0]); zonesSelectedNegate.Add(gwent.GetComponent<Gwent>().zonesSelect[1]);

}

private IEnumerator ZoneSelectedInmune(BoxInfo boxInfo)

{

*yield return* StartCoroutine(SelectZoneBox(boxInfo));

zonesSelectedInmune.Add(gwent.GetComponent<Gwent>().zonesSelect[0]); zonesSelectedInmune.Add(gwent.GetComponent<Gwent>().zonesSelect[1]);

}

private bool IsInmune(DisplayCard card)

{

List<DisplayCard> listOfCardsInmune = new();

*foreach* (var zone *in* zonesSelectedInmune)

{

*foreach* (Transform displayCard *in* zone.transform)

{

listOfCardsInmune.Add(displayCard.gameObject.GetComponent<DisplayCard>());

}

}

*return* listOfCardsInmune.Contains(card);

}