**card.cs**

Description: This script is attached to every card in the game. It stores all card effects and stats, except temporary buffs, which are handled by **EffectManager.cs** and stored in **buffs.cs**. It also handles card sounds and effects during gameplay. When the player clicks on a card in edit deck mode or main menu (promo cards), the functionality is also handled by this script.

Methods that you would likely want to modify when customizing your game:

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void UpdateCreatureAtkDefLabels () | This method gets called when a creature's attack and defense are modified (receives damage, is buffed, etc). This method updates the text and font color of attack/health icons. By default, creature stat font is blue when buffed, white when it is unchanged, red when the stat has been lowered. You can change the colors here if you want. |
| void OnMouseOver () | By default, when a player hovers the mouse over a card, it zooms in after a certain amount of time. Card zooming behavior can be changed here. Also by default, right-clicking on a creature in play opens up the activated abilities menu (if the creature has abilities). This can be changed here as well. |
| void OnMouseExit () | By default, this method unzooms the card when the mouse is no longer hovering over the card’s collider. |
| public void OnMouseDown () | Handles what happens when the user left-clicks on a card. These vary by state:   * In edit deck mode, this method initiates card movement from collection to deck or vice versa (called methods in **editdeckscripts.cs**). * In the main menu, clicking on a card means the card is a promo and this method initiates a connection to the server to buy it. * During the game, this method is primarily used to validate the target when the player chooses a target for attacking or using an effect. If it’s the player’s turn and the player has clicked a card and no target is needed, this method initiates PlayCard() to handle the card, whether it's playing it if it's in hand, adding mana if it's a land, etc.   **If you want to add a custom target type,** and it has to be chosen by player (as opposed to automatic targets, such as: “enemy player”), add a case to the switch statement in this method. |
| void BadTarget () | Called when an invalid target is selected. Displays a debug line and a player warning. Change the warning here if you don’t like the default one: “This is not a valid target for this spell” |
| public void PlayCard () | This method is called from OnMouseDown() when the player right-clicks a card. **Modify it if you need a new card type or a new game zone.** |
| IEnumerator WaitForTargetAndDoEffect (int Target, int effect\_number) | Waits for player to choose a target, then starts the effect. **If your custom target type needs a card that is not in game** (in deck, graveyard etc), add a new line here similar to targets 3, 50, 51 in the method. |
| bool ChooseAutomaticTargetsAndDoEffect (  int effect\_number, bool AI=false) | Checks if the target type is one of the automatic targets (that are not chosen by player). **If your custom effect doesn’t require choosing targets**, add a case to the switch statement here. |
| bool ValidSpell () | Called by OnMouseDown() when the player clicks a spell card that has certain requirements. Regular target validation (for when the player chooses targets) is done in OnMouseDown. This method is used to stop spells with *random* targets like “random enemy creature” from starting at all if the enemy doesn’t have a creature in play.  **If your custom effect chooses targets at random or has other requirements (such as a specific creature type in play), add a check here.** |
| public void IsAttacked (card Attacker) | Called when a creature is attacked. Plays the combat sounds and particle effects, calculates the damage received and lowers the creature’s health accordingly. **If you want to add a custom combat mechanic**, such as “triple strike”, add it here. |
| public void IsHitBySpell (Vector3 param) | param.x = damage amount  param.y = damage type  param.z = the spell card's ID  Currently there are two damage types: fire and combat. The damage type is specified to indicate what kind of special effect and how to treat the damage. For example, a creature might have an effect that causes it to only take half damage from attacks, so it is important to discern the type.  **If you want some spells to have different visual/sound effects** (for ex. lightning spells), add a damage type here. |

The following methods can be useful in your custom effects and mechanics. **I don't recommend modifying these unless absolutely needed, as it can break existing functionality.**

|  |  |
| --- | --- |
| **Method** | **Description** |
| void Kill () | Destroys the card, or moves it to graveyard if the boolean OptionGraveyard is set to ‘true’ in MainMenu. |
| bool IsACreatureInGame () | Returns ‘true’ if the card is a creature in game. This creature can belong to any player. |
| void AssignTarget (GameObject targetgameobject = null) | This is called by OnMouseDown() when a target is chosen by the player, and it adds a GameObject to the list of player targets (Player.targets). Used in switch cases in OnMouseDown() to assign a target. |
| void AssignTargets (int targets\_needed) | Same as the previous method, but when 2+ targets are needed. Check switch case 4 in this script for example usage. |
| public void Turn () | Turns the card the amount of degrees set in MainMenu.OptionTurnDegrees to indicate that it has attacked/used its ability this turn. For example, in Magic: The Gathering cards generally 'tap' (turn sideways) when using abilities or attacking. If your effect or creature ability causes this, then call this method.  If you need to check if a card has been turned, use card.isTurned == true. |
| public void UnTurn () | Returns the card that has been rotated by Turn() to its normal position. Sets card.isTurned to `false` |
| public void RestoreColor () | Restores the card's color. Use it in your visual effects if they involve changing the card color. |

**These methods are central to the kit's functionality. I don't recommend modifying the following methods or using them on their own**:

|  |  |
| --- | --- |
| **Method** | **Description** |
| void MoveToGraveyard () | Un-turns the creature if it was turned and moves it to the graveyard. Called by Kill() if the boolean OptionGraveyard is set to ‘true’ in MainMenu.  **If you want to kill a creature, use Kill() instead.** |
| void ZoomCard () | Zooms in on a card. Also temporary moves some of the zoomed cards so they don’t go out of screen borders. To change the scale of zoomed cards, set variables Zoom and ZoomEditDeckMode in **card.cs**. |
| IEnumerator UnZoomCard () | Unzooms a card |
| void SendCard () | Used to initiate multiplayer RPCs in **Logic.cs** |
| void SendEffect (int effect\_number) |
| public void SendHandCard () |
| public void PlayEnemyCardMultiplayer () | Used for “playing” the cards that the multiplayer opponent has played and sent. |
| public void PlayEnemyHandCardMultiplayer () |
| public void FromHandCreature (bool AI=false) | If your game is similar to Magic: the Gathering or Hearthstone, you don’t need to change this method. It places a creature in game when played from the hand and subtracts the creature’s mana cost from the player. |
| public void FromHandLand (bool ForPlayer=true) | Same as the previous method, but plays a land card instead of a creature. |
| IEnumerator WaitForMultiplayerTargetAndDoEffect  (int effect\_number) | Waits for the multiplayer opponent to send his or her target. In most cases, targets for effects are sent before the effect itself is sent and this method is not invoked. This is currently only invoked when a creature has an ability that triggers upon entering play, for waiting on a target after a creature's RPC is sent. For example, if a creature deals damage to a target upon entering play, this method will be invoked by the player who does not own the creature in order to wait for the opponent to select a target. This results in the player seeing the creature enter play first, and then the game waits for the opponent to select a target for the effect. |
| public void SendTargetsAndEffect  (int effect\_number) | Initiates sending of spells and creature activated abilities to the multiplayer opponent (other abilities are handled on the opponent’s client automatically). |
| public void ApplyEffect (int effect\_number,  bool AI=false) | When called, this sorts the effect by target type (single target, multiple target, player/enemy, etc.) and invokes the methods accordingly. It also determines whether the effect is being used in single player (if AI is set to `true`) or multiplayer. |
| IEnumerator PlayerSpellAfterEffects () | Destroys the spell card that has been cast or moves it to graveyard (if MainMenu.OptionGraveyard is set to ‘true’). Don’t change if your game is similar to MTG or Hearthstone, where spell cards enter the graveyard after they are played. |
| public void FromHandSpell (bool AI=false) | Checks if the spell can be cast by the player (if AI is ‘false’) and initiates the next method in the spell sequence, ApplyEffect(), if it can.  If called by the AI (AI is `true`), then ShowCardAndWait() is called. |
| IEnumerator ShowCardAndWait (bool AI) | When the opponent is playing a spell, shows the zoomed card to the player and starts doing the effect. Only change this if you need to change the position of the zoomed spell or the time for which it’s shown. |
| IEnumerator AttackTarget () | Called when a creature is attacking. Waits for the player to choose a target and calls IsAttacked on target. |
| public void CreatureAttack (bool AI=false) | Starts the attack sequence of methods depending on who is attacking. This method can be called by this client for the player, for the AI, and for a multiplayer opponent.  If it's the player on this client, this method lets the player choose his/her target. If it's the AI then the attack target is calculated. In multiplayer, the opponent has already sent the player the target for the attack and this will be called to resolve it. |
| public bool IsCriticalStrike () | Calculates if this attack is a critical strike. |
| public void IsHealed (int param) | Plays the healing sounds and particle effects, and increases this card's health by param. Note that this card's original health cannot be exceeded when this method is called. |
| IEnumerator IsDealtSpellDamage (int amount,  int cardid) | Waits for 0.8 seconds (can be changed) and deals spell damage to this card equal to amount.  cardid is the unique ID for the card in-game. Note that this is *not* the card ID in the .xml database. |