Progetto di Realtà Aumentata

Valentina Ferraioli

Settembre 2020

Nota: Per l'utilizzo dell'app, scaricare i marker da Qui

1 Introduzione

ChimicaLAB è un applicazione Android pensata per introdurre la tecnologia della realtà aumentata all'intero dell'ambiente didattico, in particolare, per l'insegnamento della chimica.
L'applicazione non ha lo scopo di sostituirsi all'insegnante o al libro di testo, bensì di essere un supporto a questi. L'applicazione, infatti, è stata pensata durante un anno particolare a causa del emergenza sanitaria dettata dal covid-19. L' idea è nata vedendo che durante la DAD nelle scuole superiori, una parte fondamentale dell'insegnamento della chimica, ovvero l' esperienza in laboratorio, è venuta completamente a mancare.
Questo perché gli esperimenti chimici richiedono di essere svolti in ambienti adeguati, sotto la supervisione di un adulto, e con sostanze potenzialmente pericolose.
Tutti questi requisiti non possono essere garantiti nello spazio domestico, ma talvolta può succedere che anche a scuola non sia possibile fare queste esperienze, per assenza dello studente o mancanza delle strutture adeguate.
Quest'applicazione potenzialmente può trattare l'intero piano didattico. Alcuni degli argomenti da trattare sono inseriti nella pagina iniziale, ma per il momento è stato sviluppato in modo completo solo un capitolo specifico, che è quello delle reazioni chimiche. In questo capitolo, viene fornito un ripasso teorico sull'argomento, in formato pdf o video, e gli strumenti per procedere con la simulazione degli esperimenti correlati, attraverso AR. Per la simulazione degli esperimenti è stata creata un'applicazione AR marker-based, che consiste nell'identificazione dei marker come sostanze chimiche, e la manifestazione di una reazione chimica quando avviene un urto tra due marker. Nelle sezioni successive andremo a vedere nello specifico i casi d'uso, l'architettura dell'applicazione, possibili criticità e sviluppi futuri.

2 Casi d'Uso

In questa sezione definiamo i requisiti funzionali del sistema sviluppando in modo approfondito i casi d'uso dalla prospettiva dell'utente.

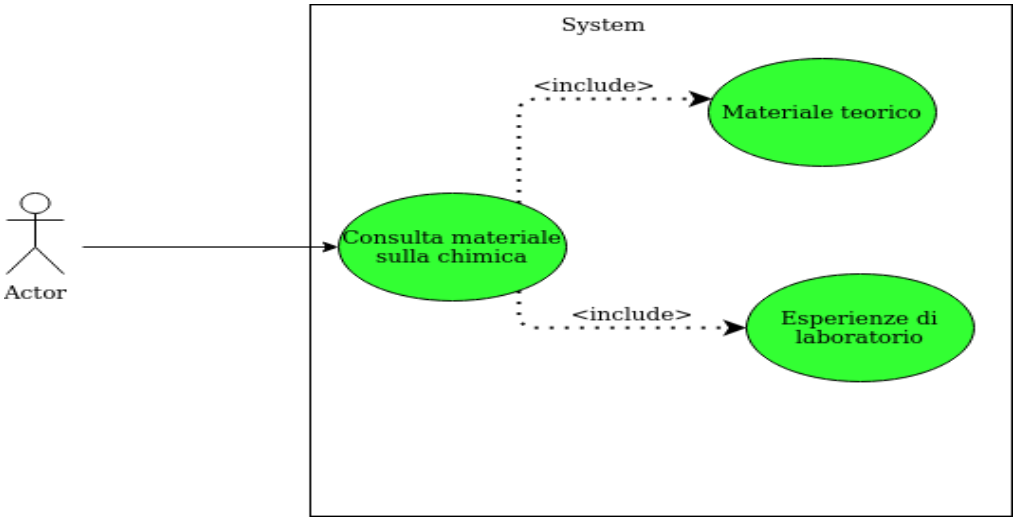


Figura 1

Per comprendere meglio i casi d'uso in seguito definiamo Obiettivi, Attori, e il comportamento del sistema.

Obiettivo

Lo studente può eseguire due tipo di task:

- studiare la parte teorica relativa all'argomento reazioni chimiche.
- simulare un esperienza di laboratorio, nel quale è possibile osservare come si manifestano le reazioni chimiche.

Attore

Dagli obiettivi emerge che l'attore principale di questa applicazione sono gli studenti. Nello specifico lo user target di questo tipo di applicazione sono i ragazzi del secondo biennio di un istituto superiore, nel quale è previsto l'insegnamento della chimica. L'applicazione, come detto precedentemente fa da supporto ai libri di testo e/o alle insegnanti, per questo si suppone ne venga fatto riferimento all'interno del libro di testo, o che venga citata dagli insegnanti.

Sistema

A questo punto, dopo aver identificato attori e obiettivi possiamo procedere osservando il comportamento del sistema. Per fare ciò andiamo a sviluppare un diagramma di flusso che rappresenta l'insieme di interazioni che devono avvenire tra utente e sistema per arrivare all'obiettivo. Si vedrà un possibile percorso, ma se ne potranno fare altri alternativi.

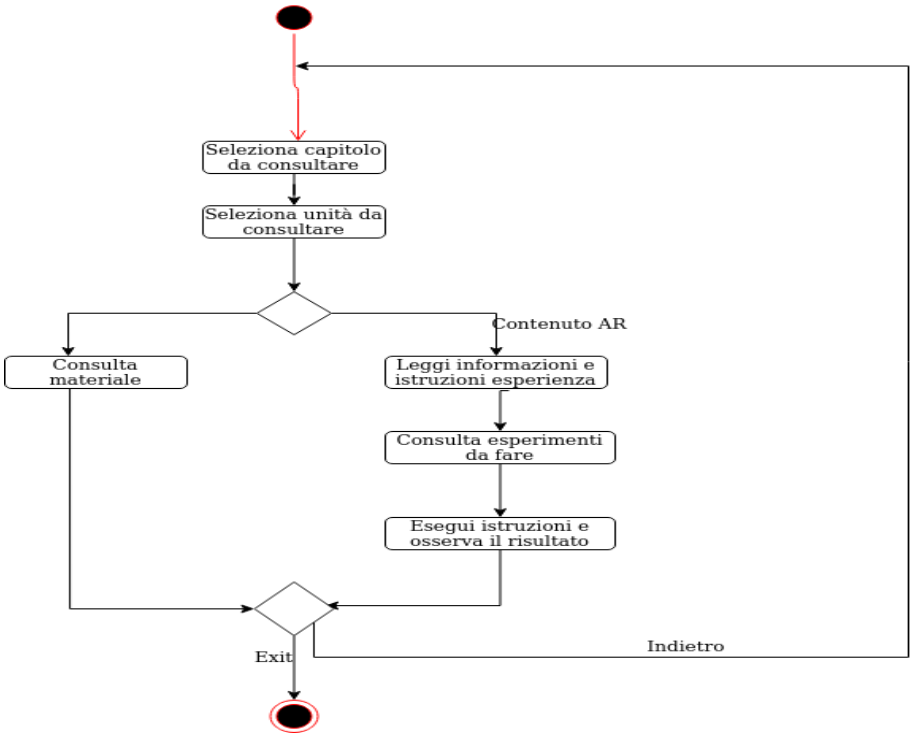


Figura 2: Event Flow Chart per il completamento di un task

3 Architettura dell'Applicazione

Per l'applicazione è stata utilizzata la tradizionale architettura Android formata da 5 livelli:

- Linux Kernel
- Hardware Abstraction Layer
- Libreria e Android Runtime
- Java API Framework
- System Apps

Per la creazione dell'applicazione si è agito soltanto al livello **Libreria**. dove è stata inclusa una libreria esterna, chiamata *unityLibrary*, che costituisce il fulcro di quest'applicazione.
La libreria in questione contiene l'ambiente di realtà aumentata nel quale è possibile procedere con la simulazione degli esperimenti. È stata sviluppata in *Unity*, e fa uso della libreria Vuforia per il riconoscimento e il tracciamento delle immagini.
Per procedere con l'esperimento, la libreria unityLibrary ha lo scopo di riconoscere due marker e posizionare su di essi due modelli 3D, che rappresentano una provetta contenente la sostanza indicata dal marker. Successivamente, dovrà rilevare eventuali urti tra i due marker che daranno luogo alla reazione chimica. La reazione si manifesta con la creazione di uno modello 3D, rappresentante il prodotto ottenuto dal contatto tra i due reagenti (i due marker iniziali). Oltre al prodotto della reazione saranno fornite anche alcune informazioni, come il tipo di reazione che sta avvenendo e la sua equazione chimica.

La creazione del prodotto viene eseguito attraverso lo script *createreaction.cs* (chemistry/Assets/Script/create_reaction.cs), il quale è stato assegnato ai gameobject rappresentanti i marker. Ogni Marker è stato definito con un boxCollider e rigidBody, affinchè la collisione potesse essere rilevata dallo script. Una volta rilevata la collisione vengono identificati i marker coinvolti e viene instanziato dinamicamente il prodotto. Di seguito un pezzo del codice presente in *creareReaction.cs*, che esegue quanto descritto precedentemente.

```
void OnCollisionEnter(Collision collision){
    if(allPreviousCollisionEnded==true){
        //getting and storing initial reagents name
        tmpReagentName1 =
        gameObject.GetComponentInChildren (typeof(TextMesh)) as TextMesh;
        tmpReagentName2 =
        collision.gameObject.GetComponentInChildren (typeof(TextMesh)) as TextMesh;
        reagentName1 = tmpReagentName1.text;
        reagentName2 = tmpReagentName2.text;

        //getting collision point, which will be used for positioning
        the result product
        Vector3 contactPosition=collision.contacts[0].point;
        Quaternion rot = transform.rotation;
        Vector3 reactionResultPosition =
        new Vector3(contactPosition.x+0.5f, contactPosition.y, contactPosition.z);
        Vector3 infoPosition =
        new Vector3(contactPosition.x-0.5f, contactPosition.y-0.4f, contactPosition.z);

        //identify the marker which are colliding
        if(gameObject.name == "Zinc_card"){
            //instantiate the product flask and instantiate the correct stoichiometric
            formula
            if(collision.gameObject.name == "Hydrochloric_card"){
                tmpReagentName2.text = "2HCl"; product=Instantiate(Resources.Load(
                "zinc_chloride", typeof(GameObject)), reactionResultPosition , rot,
                transform.parent) as GameObject;
```

```
                    collisionHappens = true;
                    ReactionType = "Reazione di Scambio" ;

                }
                }
                TextMesh productName =
                product.GetComponentInChildren (typeof(TextMesh)) as TextMesh;
            //Text MEsh for Reaction information
            tempTextBox = (GameObject)Instantiate(TextReaction, infoPosition, rot);

            //Grabs the TextMesh component from the game object
            Text theText =
            tempTextBox.transform.GetComponentInChildren (typeof(Text)) as Text;
            //Sets the text.
            theText.text =
            ReactionType + "\n" + tmpReagentName1.text + " + "+ tmpReagentName2.text + " -> "

            allPreviousCollisionEnded = false;
        }
    }
}
```

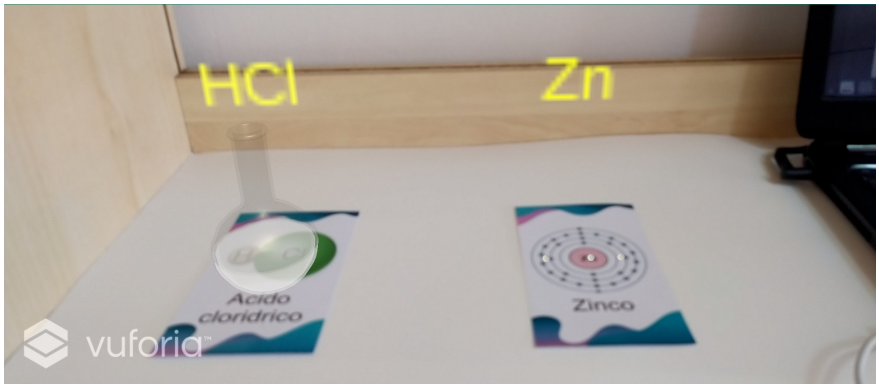


Figura 3: Identificazione dei marker

Figura 4: rilevazione di una collisione tra i due marker, con conseguente instanziazione del prodotto.

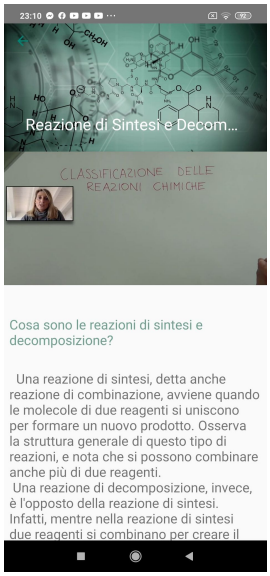
Inoltre, ai diversi prefab rappresentanti i prodotti sono stati assegnati degli script per la gestione delle animazione, se necessario . Ad esempio nella reazione "Zinco + Solfato di Rame ", per attivare il cambiamento di colore dello zinco, dopo essere entrato in contatto con il solfato di rame viene utilizzato lo script *changecolor.cs*.

```
    public class change_color : MonoBehaviour
{
    MeshRenderer myRenderer;
    float speed=0.1f;
    Color startColor ;
    Color endColor = Color.black;
    // Start is called before the first frame update
    void Start()
    {
        myRenderer = GetComponent<MeshRenderer>();
        startColor = myRenderer.material.color;

        //myRenderer.material.color =
        Color.Lerp(myRenderer.material.color, Color.red, 0.3f);
        StartCoroutine(ChangeEngineColour());
    }

    private IEnumerator ChangeEngineColour(){
    float tick = 0f;
    while (myRenderer.material.color != endColor){
        tick += Time.deltaTime * speed;
        myRenderer.material.color = Color.Lerp(startColor, endColor, tick);
        yield return null;
    }
}
}
```

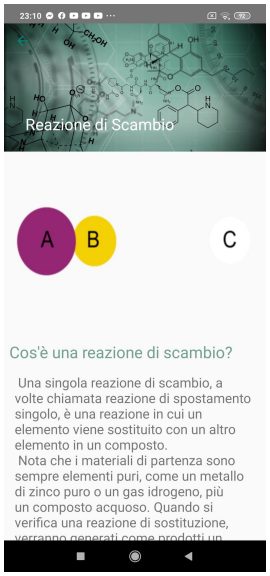
Dal livello **Java API Framework** , invece si è fatto uso degli elementi per la creazione dell'applicazione, come l'Activity Manager, il View Manager e il Resource Manager.

L'applicazione finale è formata da 4 Activity: Home, Teoria, IstruzioniEsperimento, UnityPlayer.

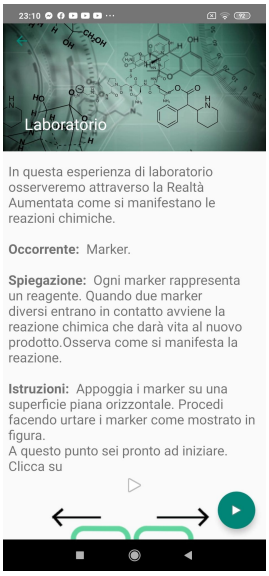
(a) Home



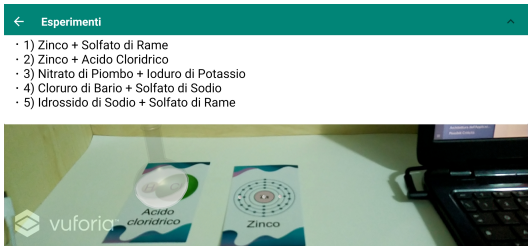
(b) Teoria-video



(c) Teoria - testo



(d) Istruzioni



(e) UnityPlayer

Figura 5

In generale le activities sono molto basilari.Vediamo che l'activity Home è

composta da un ExpandableListView per la visualizzazioni degli argomenti trattati e i paragrafi presenti per ognuno. Ogni paragrafo è preceduto da un'icona che indica se si tratta di teoria, che potrà includere un contenuto testuale o video, oppure di un'esperienza AR. Nel caso in cui si scelga un paragrafo contente un'esperienza AR, allora si accede all'activity unityplayer, che è quella il cuore dell'applicazione. Questa activity è composta da diverse text view che vanno a formare una toolbar espandibile, dal quale è possibile leggere gli esperimenti che è possibile fare. Al momento della creazione dell'activity, nel metodo OnCreate() viene aggiunto lo UnityPlayer presente nella libreria unityLibrary importata precedentemente.

4 Possibili Criticità

ChimicaLAB è stata progettata esclusivamente per Smartphone con sistema operativo Android, inoltre richiede come versione minima Android 6.0.
Sebbene la versione minima richiesta sia soddisfatta, l'applicazione potrebbe presentare dei problemi dovuti al supporto di Vuforia da parte del device. Qui si può trovare la lista di device che supportano Vuforia.

5 Sviluppi Futuri

L'applicazione come abbiamo visto è stata impostata per coprire l'intero programma didattico. Tuttavia è stato sviluppato in modo completo soltanto il capitolo relativo alle reazioni chimiche. Per cui, un possibile sviluppo naturale dell'applicazione sarebbe lo sviluppo di tutti gli argomenti previsti, con relative esperienze in AR. Inoltre, affinchè diventi uno strumento di studio completo si potrebbero inserire dei paragrafi in cui avviene una verifica delle nozioni apprese, i quali potrebbero essere pensati utilizzando la tecnica del quiz, nel quale potrebbero essere pensate domande in cui sia necessario fare uso di AR. Una possibile domanda di questo tipo, potrebbe essere: "Indica quali reagenti formano un precipitato", nel quale si prevede che i reagenti debbano essere identificati con AR.