

Зміст

1	Віртуальна хімічна лабораторія у доповненій реальності: від ідеї до реалізації	2
1.1	Ідея	2
1.2	Підготовка маркерів, зображень та відео	2
1.3	Програмна реалізація	7
1.3.1	Початкова сторінка	7
1.3.2	Реалізація віртуальної лабораторії	7
1.3.3	Розгортання програмного забезпечення	15
1.3.4	Тестування віртуальної лабораторії	16
A	Файл index.html	16
B	Файл ar.html	17

1 Віртуальна хімічна лабораторія у доповненій реальності: від ідеї до реалізації

1.1 Ідея

Загальна: створити віртуальну хімічну лабораторію, дії в якій над віртуальними об'єктами, заданими зображеннями або моделями реальних об'єктів, приводять до відтворення процесу їх еталонної взаємодії.

Частинна: для набору реактивів, необхідних для виконання лабораторної роботи “Якісні реакції хлорид-, бромід- та йодид-йонів”, надати учням можливість випробувати всі варіанти їх поєднання.





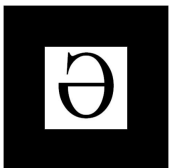

Для реалізації даної ідеї пропонується кожний із реактивів співставити із маркером доповненої реальності. Для “зливання” реактивів два маркери необхідно наблизити один до одного й, по досягненню певної відстані, відобразити відеозапис реального експерименту.







1.2 Підготовка маркерів, зображень та відео







Для підготовки маркерів необхідно обрати засіб компіляції зображень та виділення їх опорних точок, що відповідає застосовуваному рушію доповненої реальності. Так, при виборі в якості рушія AR.js таким компілятором буде “AR.js Marker Training” (<https://ar-js-org.github.io/AR.js/three.js/examples/marker-training/examples/generator.html>), що генерує кольорові квадратні маркери роздільною здатністю 16×16 у форматі .patt: 3 кольорові площини (червона, зелена та синя) і 4 орієнтаціях (кути повороту 0° , 90° , 180° та 270°).



Згенеровані маркери пронумеровані від 0 до 9 та співставлені із назвами і зображення реактивів (табл. 1.1).

Таблиця відповідності реактивів, маркерів та зображень.

№	Реактив	Маркер	Зображення
0	AgNO_3	pattern-0.patt 	AgNO3.jpg 
1	$\text{Pb}(\text{NO}_3)_2$	pattern-1.patt 	Pb(NO3)2.jpg 
2	HCl	pattern-2.patt 	HCl.jpg 

№	Реактив	Маркер	Зображення
3	KCl	pattern-3.patt 	KCl.jpg 
4	NaCl	pattern-4.patt 	NaCl.jpg 
5	NH ₄ Cl	pattern-5.patt 	NH ₄ Cl.jpg 

№	Реактив	Маркер	Зображення
6	KBr	pattern-6.patt 	KBr.jpg 
7	NH ₄ Br	pattern-7.patt 	NH4Br.jpg 
8	KI	pattern-8.patt 	KI.jpg 

№	Реактив	Маркер	Зображення
9	NH_4I	pattern-9.patt 	NH4I.jpg 

Попарне застосування 10 маркерів без урахування порядку комбінування дає такі 45 комбінацій:

0-1, 0-2, 0-3, 0-4, 0-5, 0-6, 0-7, 0-8, 0-9

1-2, 1-3, 1-4, 1-5, 1-6, 1-7, 1-8, 1-9

2-3, 2-4, 2-5, 2-6, 2-7, 2-8, 2-9

3-4, 3-5, 3-6, 3-7, 3-8, 3-9

4-5, 4-6, 4-7, 4-8, 4-9

5-6, 5-7, 5-8, 5-9

6-7, 6-8, 6-9

7-8, 7-9

8-9

Далеко не всіх з них мають сенс, тому відеозапис було виконано лише для тих комбінацій реактивів, в яких наявні ознаки хімічної реакції різняться (табл. 1.2).

Для випадків, коли реакція не відбувається, були додатково записані файли 09.mp4, 14.mp4, 1.1.mp4, 1.2.mp4, 1.3.mp4, 1.4.mp4, 1.5.mp4.

Створені файли маркерів, зображень та відео були завантажені у репозитарій GitHub за посиланням <https://github.com/Lefoxi/ARdip> до каталогів markers, images та video відповідно. Додатково для зручності використання зображення маркерів були зібрані у файлі markers.pdf (<https://github.com/Lefoxi/ARdip/blob/main/markers.pdf>).

Розподіл відеозаписів по маркерах.

		№	Реактив	№	Реактив
№	Реактив	0	AgNO ₃	1	Pb(NO ₃) ₂
2	HCl	16.mp4		08.mp4	
3	KCl	10.mp4		11.mp4	
4	NaCl	10.mp4		13.mp4	
5	NH ₄ Cl	17.mp4		07.mp4	
6	KBr	06.mp4		05.mp4	
7	NH ₄ Br	03.mp4		04.mp4	
8	KI	02.mp4		01.mp4	
9	NH ₄ I	02.mp4		01.mp4	

1.3 Програмна реалізація

1.3.1 Початкова сторінка

Початкова сторінка – файл index.html (додаток А), тіло якого містить два гіперпосилання:

- ar.html – власне код віртуальної лабораторії;
- markers.pdf – файл PDF із зображеннями маркерів для друку.

1.3.2 Реалізація віртуальної лабораторії

Для реалізації лабораторії було обрано бібліотеку A-Frame для створення віртуальних об'єктів та бібліотеку AR.js для їх зв'язування зі створеними маркерами.

Загальна структура файлу ar.html:

```
<html>
  <head>
    <!-- налаштування заголовку вікна -->
    <title>
      Віртуальна хімічна лабораторія у доповненій реальності
    </title>
```

```

<!-- підключення бібліотеки A-Frame -->
<script
  src="https://aframe.io/releases/1.3.0/aframe.min.js"></script>
<!-- підключення бібліотеки AR.js для роботи з A-Frame -->
<script src=
"https://raw.githubusercontent.com/AR-js-org/AR.js/master/aframe/build/aframe-ar
"></script>
<script>
  <!-- основна програма -->
</script>
</head>

<body>
  <!--Створення сцени A-Frame для роботи у доповненій реальності-->
  <a-scene vr-mode-ui="enabled: false;"
renderer="logarithmicDepthBuffer: true;" embedded
arjs="trackingMethod: best; sourceType: webcam; debugUIEnabled: false;
  >
  <!-- Наповнення сцени A-Frame -->
</a-scene>
</body>
</html>

```

Сцену A-Frame складають 4 блоки:

- 1) `<a-assets>...<a-assets>` – визначення відеотекстур;
- 2) `<a-marker>...</a-marker>` – визначення маркерів та прив'язка до них зображень і відео;
- 3) `<a-entity camera></a-entity>` – розміщення на сцені камери;
- 4) `<a-entity run></a-entity>` – звернення до компоненту `run`, визначеного у основній програмі.

Блок визначення відеотекстур складають записи виду:


```
<video id="video02" preload="auto" muted poster="images/starting.jpg"
      loop="false" src="video/16.mp4"></video>
```

- **id** – ідентифікатор відео, що складається з імені **video** та номерів маркерів, що суміщуються (згідно табл. 1.2). Якщо одне й те саме відео застосовується для декількох комбінацій маркерів, ідентифікатор набуває вигляду **video03-04**. Для випадків, коли реакція не відбувається, передбачені відеотекстури **video00-1 ... video00-7**;
- встановлення **preload** у **auto** визначає необхідність завантаження відео до першого звернення до нього – це збільшує початковий час завантаження програми, проте зменшує час, необхідний для появи відео і убезпечує від зупинок відео через нестабільне мережне з'єднання;
- **muted** – вказує на необхідність вимкнення звуку, якщо звукова доріжка наявна у відео;
- **poster** містить посилання на файл зображення, що з'являється, якщо звернення до відео відбулось до того, як воно повністю завантажилось;
- **src** – посилання на відеофайл.

Блок визначення маркерів складають 9 записів виду:

```
<a-marker type="pattern" url="markers/pattern-0.patt" id="M0"
      registerevents>
  <!-- Визначення зображення -->
  <!-- Визначення відео -->
</a-marker>
```

Кожен маркер визначається номером від 0 до 9 (згідно табл. 1.1), який входить до складу імені маркера (**pattern-0.patt** – файл опорних точок маркеру 0) та його ідентифікатора **id** (**M0** – ідентифікатора маркеру 0). До кожного маркера застосовується компонент **registerevents**, визначений в основній програмі.

Визначення зображення, що накладається на маркер після його виявлення, виконується командою виду:

```
<a-plane src="images/AgN03.jpg" rotation="-90 90 0" id="draw0">
</a-plane>
```

a-plane визначає текстуровану площину. Параметр **src** містить посилання на файл текстури – зображення реактиву згідно табл. 1.1. Через те, що зображення збережено у альбомному розташуванні, а демонструється у портретному, виконується його поворот на кут 90° навколо вісі OY . Поворот на 270° (-90°) навколо вісі OX необхідний через те, що координатний простір A-Frame повернутий на відповідний кут відносно координатного простору камери. **id** – ідентифікатор площини із зображенням, що складається зі слова **draw** та номера маркера.

До кожного маркера прив'язується від 9 (маркер 0) до 0 (маркер 9) площин із відео згідно визначеної вище схеми комбінацій маркерів:

```
<a-entity
  material="shader: flat; src: #video00-1"
  geometry="primitive: plane; width: 0.90; height: 1.60;"
  position="0 0 0" scale="1.5 1.5 1.5" rotation="-90 0 0"
  id="drawX01" visible="false">
</a-entity>
```

Співвідношення сторін площини задається її шириною **width** та висотою **height** (вказуються відносно розміру маркера, що приймається за 1). Властивість **src** матеріалу площини містить посилання на раніше визначену та завантажену відеотекстуру. Параметр **position** визначає координати центру площини (0 0 0 – співпадіння з координатами центру маркера), **scale** – півторократне масштабування (1.5 1.5 1.5) відеотекстури порівняно із визначеними шириною та висотою, **rotation** визначає поворот відповідний кут (аналогічно до повороту зображення). Ідентифікатор об'єкту **id** складається зі слова **drawX** та номерів маркерів, що зближаються, впорядкованих за зростанням.

Основна програма складається з 3 блоків:

- 1) визначення змінних:

- **howmuch** – емпірично дібране значення граничної відстані між маркерами, менше за яку вважатимемо, що має розпочатись реакція (за замовчанням 1.4);
- **isReaction** – прапорець, що вказує на те, що реакція розпочалась: якщо він встановлений, демонструється відповідний відеозапис реакції (початкове значення **false**);
- **distance** – поточна відстань між парою маркерів, що зближуються (початкове значення **howmuch+1**);
- **markerVisible** – масив станів видимості маркерів: якщо маркер видимий, значення елемента масиву з відповідним маркеру номером встановлюється у **true** (початкове значення для кожного маркера **false**);
- **M** – масив для збереження маркерів (визначених у документі HTML об'єктів **a-marker**);
- **d** – масив для збереження зображень, що накладаються на маркери (визначених у документі HTML об'єктів **a-plane**, ідентифіковані як **drawN**, де **N** – номер маркера);
- **X** – масив для збереження відео, що накладаються на маркери (визначених у документі HTML об'єктів **a-plane**, ідентифіковані як **drawXAB**, де **A**, **B** – номери маркерів, що зближуються);
- **p** – масив координат центрів маркерів, значення якого використовуються для вимірювання відстаней;
- **isVideoPlay** – прапорець, що вказує на те, що відтворюється відеозапис реакції (початкове значення **false**);

2) реєстрація компоненту **registerevents**;

3) реєстрація компоненту **run**.

Компонент **registerevents** призначений для відстеження двох подій – виявлення (**markerFound**) та втрати (**markerLost**) маркера. Після реєстрації метод **init** компоненту застосовується до усіх маркерів, до яких був доданий компонент **registerevents**, з метою визначення номеру маркера **index** та встановлення при виявленні чи знятті при втраті прапорця

його видимості у масиві `markerVisible`. При виявленні маркеру пов'язане із ним зображення `d[index]` робиться видимим, а при втраті – невидимим встановлення у відповідне значення властивості `visible`:

```
AFRAME.registerComponent("registerevents", {
  init: function () {
    let marker = this.el;
    marker.addEventListener('markerFound', function() {
      index = parseInt(marker.id[1])
      markerVisible[ index ] = true;
      d[index] = document.querySelector("#draw"+index);
      if(d[index]!==null)
        d[index].setAttribute("visible", "true");
    });
    marker.addEventListener('markerLost', function() {
      index = parseInt(marker.id[1])
      markerVisible[ index ] = false;
      if(d[index]!==null)
        d[index].setAttribute("visible", "false");
    });
  }
});
```

Компонент `run` не відповідає жодному видимому елементу – він виконує загальне управління сценою, для чого реєструється як одноразово виконуваний метод `init`, так й виконуваний за таймером (бажано для кожного кадру) метод `tick`:

```
AFRAME.registerComponent("run", {
  init: function() {
    // ініціалізація компоненту
  },

  tick: function (time, deltaTime) {
    // метод, що викликається за таймером
  }
});
```

```
});
```

У методі `init` заповнюються визначені раніше масиви: `M` – посиланнями на маркери, `d` – посиланнями на зображення реактивів (відображаються для маркерів, що є видимими), `X` – посиланнями на прив'язане до маркерів відео, та `p` – нульовими координатними векторами:

```
for (let i = 0; i < 10 ; i++) {  
    M[i] = document.querySelector("#M"+i);  
    d[i] = document.querySelector("#draw"+i);  
    p[i] = new THREE.Vector3();  
    if (markerVisible [i])  
        d[i].setAttribute("visible", "true");  
    for (let j = 0; j < 10 ; j++)  
        if(i!=j)  
            X[i][j] = document.querySelector("#drawX"+i+""+j);  
}
```

Метод `tick` постійно відслідковує видимість та взаємне розташування маркерів:

- 1) до масиву `visible` заносимо номери маркерів, що у поточний момент видимі – для цього аналізуємо зміст масиву `markerVisible`, встановлений компонентом `registerevents`:

```
let visible = [] ;  
for (let i=0;i<10;i++)  
    if (markerVisible[i])  
        visible.push(i);
```

- 2) будемо вважати, що для реакції необхідні рівно 2 реактиви, тому випадки, коли видимими є менше двох маркерів (або більше двох), ігноруватимемо – якщо демонструвалось відео, робимо його невидимим та встановлюємо прапорець `isVideoPlay` у `false`:

```

if(visible.length!=2) {
    for (let i = 0; i < 10 ; i++)
        for (let j = 0; j < 10 ; j++)
            if(i!=j && X[i][j]!=null)
                X[i][j].setAttribute("visible", "false");
    isVideoPlay = false;
    return;
}

```

3) визначаємо номери двох видимих маркерів – `marker1` та `marker2`:

```

marker1 = visible[0];
marker2 = visible[1];

```

4) визначаємо координати маркерів та зберігаємо їх у відповідних елементах координатного масиву `p`:

```

M[marker1].object3D.getWorldPosition(p[marker1]);
M[marker2].object3D.getWorldPosition(p[marker2]);

```

5) знаходимо відстань між маркерами:

```

distance = p[marker1].distanceTo( p[marker2] );

```

6) у прапорці `isReaction` встановлюємо, чи достатньо близькі маркери для перебігу реакції:

```

isReaction=(distance <= howmuch);

```

7) прапорець `isReaction` не вказує, чи йшла реакція до поточного моменту – визначити це можна, проаналізувавши, чи вже програвалось відео: якщо ні, знімаємо з обох маркерів зображення, робимо видимим відповідне відео та розпочинаємо його програвати із самого початку:

```

if(isReaction) {
    if(!isVideoPlay) {

```

```

        if(d[marker1]!==null)
            d[marker1].setAttribute("visible", "false");
        if(d[marker2]!==null)
            d[marker2].setAttribute("visible", "false");
        X[marker1][marker2].setAttribute("visible", "true");
        var id="#" + X[marker1][marker2]
            .getAttribute("material").src.getAttribute("id");
        var video=document.querySelector(id);
        video.currentTime=0;
        video.play();
        isVideoPlay = true;
    }
}

```

- 8) якщо маркери віддалились на відстань, що відповідає припиненню реакцію, призупиняємо відео, робимо його невидимим та знову накладємо зображення реактивів на обидва маркери:

```

else {
    if(d[marker1]!==null)
        d[marker1].setAttribute("visible", "true");
    if(d[marker2]!==null)
        d[marker2].setAttribute("visible", "true");
    X[marker1][marker2].setAttribute("visible", "false");
    var id="#" + X[marker1][marker2]
        .getAttribute("material").src.getAttribute("id");
    var video=document.querySelector(id);
    video.currentTime=0;
    video.pause();
    isVideoPlay = false;
}

```

1.3.3 Розгортання програмного забезпечення

Ураховуючи, що вихідні тексти програмного забезпечення було розміщені у репозитарії GitHub, його розгортання було виконано на

сторінках GitHub (GitHub Pages). Для цього у налаштуваннях репозитарію (Settings – Pages) була обрана головна гілка репозитарію (main) та його кореневий каталог, що містить початкову сторінку (файл index.html), та виконано збереження (Save). Після цього виконується процедура розгортання програмного забезпечення (публікація сайт) за посиланням <https://lefoxi.github.io/ARdip/>. Наявність на опублікованому сайті файлу index.html надає можливість автоматично перейти до початкової сторінки після переходу за вказаним вище посиланням.

Для оновлення розгорнутого програмного забезпечення достатньо виконати завантажити змінені файли до репозитарію.

1.3.4 Тестування віртуальної лабораторії

... (*дали – самі*)

А Файл index.html

```
<html>
  <head>
    <style>
      button {
        color: lightgreen; background: #0000aa; height: 150px;
        width: 450px; font-size:200%; border: 2px solid powderblue;
        padding: 30px; position: fixed; padding: 1vh;
        bottom: 1vh; left: 50%; transform: translate(-50%, -50%);
        top: 50%;
      }
      #markers {
        font-size: 138%; border: 2px solid powderblue; padding: 30px;
        position: fixed; bottom: 1vh; left: 50%;
        transform: translateX(-50%);
      }
    </style>
  </head>
  <body>
```



```
<a href="ar.html" target="_blank"><button>Перейти до віртуальної  
лабораторії</button></a><p>  
<a href="markers.pdf" target="_blank" id="markers">Натисніть для  
завантаження набору маркерів - роздрукуйте та розріжте їх. У  
віртуальній лабораторії покажіть їх камері, й, після появи на  
маркері зображення, спробуйте їх зблизити, але без накладання.</a>  
</body>  
</html>
```

Б Файл ar.html

Тут слід розмістити код відповідного файлу.