# Анотація

# Зміст

[1 Анотація 1](#_Toc453758892)

[2 Зміст 2](#_Toc453758893)

[3 Короткий словник основних термінів 5](#_Toc453758894)

[4 Вступ 6](#_Toc453758895)

[4.1 Огляд та аналіз сучасного стану 6](#_Toc453758896)

[4.1.1 Сучасний стан Natural User Interface (NUI) 6](#_Toc453758897)

[4.2 Обгрунтування актуальності теми і мети роботи 6](#_Toc453758898)

[5 Аналіз 8](#_Toc453758899)

[5.1 Аналіз поведінки користувача 8](#_Toc453758900)

[5.1.1 Заохочення користувача почати роботу 8](#_Toc453758901)

[6 Вибір засобів реалізації 10](#_Toc453758902)

[6.1 Датчики жестів NUI 10](#_Toc453758903)

[6.2 Бібліотеки виявлення жестів 11](#_Toc453758904)

[6.3 Платформа для реалізації презентаційного інтерфейсу 12](#_Toc453758905)

[6.4 Комунікація реального режиму часу між інтерфейсом користувача і обробником Kinect 13](#_Toc453758906)

[6.5 Комунікація в режимі підписки між сервером інсталяції і сервером статистики 15](#_Toc453758907)

[7 РеалізаціЯ 16](#_Toc453758908)

[7.1 Основні актори та use-case системи 16](#_Toc453758909)

[7.2 Компоненти системи 17](#_Toc453758910)

[7.3 Взаємодія між компонентами 19](#_Toc453758911)

[7.4 Технічний стек 20](#_Toc453758912)

[7.4.1 Технічний стек Presenter UI 20](#_Toc453758913)

[7.4.2 Технічний стек KinectGlue 21](#_Toc453758914)

[7.4.3 Технічний стек Admin UI 22](#_Toc453758915)

[7.4.4 Технічний стек StatsService 23](#_Toc453758916)

[7.5 Внутрішня організація системи 24](#_Toc453758917)

[7.5.1 Реактивна комунікація між компонентами серверу інсталяції (KinectGlue) 24](#_Toc453758918)

[7.5.2 Організація реалізації емітерів, виявлення рухів 25](#_Toc453758919)

[7.6 Алгоритми виявлення жестів 28](#_Toc453758920)

[7.6.1 Короткі загальні відомості 28](#_Toc453758921)

[7.7 Виявлення активного користувача 29](#_Toc453758922)

[7.8 Загальне бачення евристичних алгоритмів 30](#_Toc453758923)

[7.9 Виявлення руху «перегортання» 32](#_Toc453758924)

[7.10 Концепція фізичної інсталяції 34](#_Toc453758925)

[7.11 Результати роботи системи 35](#_Toc453758926)

[7.11.1 Інтерфейс користувача 35](#_Toc453758927)

[7.11.2 Інтерфейс адміністратора 37](#_Toc453758928)

[8 Висновки 38](#_Toc453758929)

[9 Список використаних джерел 39](#_Toc453758930)

[10 Додатки 40](#_Toc453758931)

# Короткий словник основних термінів

Natural User Interface (NUI) - користувальницький інтерфейс, в якому використовуються природня поведінка користувача, як то жести, розмова, положення тіла, тощо.

# Вступ

## Огляд та аналіз сучасного стану

### Сучасний стан Natural User Interface (NUI)

NUI дозволяють користувачам взаємодіяти з пристроєм за допомогою дій, які є природними, інтуїтивно зрозумілим і загальним для повсякденного людської поведінки. Розрізняють багато різних варіантів таких як: керування жестами, голосом, переміщенням тіла, керування дотиком, тощо.

Поточне рішення використовує керування за допомогою жестів, а також переміщення тіла. Тому розглянемо сучасний стан NUI в цих областях.

Для розпізнавання жестів та переміщення тіла використовуються датчики які базуються на скануванні відбиття інфрачервоної мережі генерованої інфрачервоними джерелом. Це дає можливість отримати інформації про відстань до кожного пікселя. На основі інформації про відстань, використовуючи алгоритми машинного навчання такі як Random Forest класифікатор виділяє тіло на загальному плані. На наступному етапі методами машинного навчання розпізнаються конкретні частини тіла. Ця інформація стає доступною програмному забезпеченню, яке вже має можливість розпізнавати жести та рухи.

## Обгрунтування актуальності теми і мети роботи

NUI находиться на етапі ранньої адоптації по класифікації technology adoption life cycle, це означає що дуже вірогідно що розвиток цього напряму буде сильно прискорюватись найближчі роки. Фактично зараз проходить етап пошуку шаблонів, повторюваних рішень і можливих сценаріїв використання. На наступному етапі NUI почне виходити на масового споживача, і відвойовувати сектори які зараз зайняті GUI.

Тема бакалаврські роботи відкриває можливість розробки алгоритмів NUI на прикладі керування інтерфейсом загального користування. Розроблене рішення було протестоване на реальній інсталяції з реальними користувачами.

# Аналіз

## Аналіз поведінки користувача

Для реалізації NUI було проаналізовано досвід користувача і розроблено зв’язану систему рухів що дозволяє інтуїтивно виконати задачі.

Для аналізу рухів були враховані такі фактори

Інтуїтивність – рух має бути інутїтивним, очікуванним, має мати мінімальне навчання. Якщо можливо повторювати рухи в звичайному оточенні.

Точність спрацьовування – не всі рухи можна виділити на фоні інших рухів.

Відсутність конфілктів – багато рухів повністю чи частково конфліктують між собою. Важливо отримати схему рухів без конфліктів.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Мотивація | Інтуїтивність | Точність | Конфлікти |
| Відповідь загального плану на рух користувача | Заохочення користувача спробувати | Висока | Не потребується | Ні |
| Змін |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

### Заохочення користувача почати роботу

Для користувача перше знайомство з NUI це один з найважливіших етапів. Користувач не бачить і не чує NUI, на початку для користувача системи не існує взагалі. Для того щоб вирішити цю проблему було вирішено використовувати декілька варіантів заохочення.

#### Відповідь інтерфейсу на рух користувача

Коли потенційний користувач опиняється в зоні детектування датчиком, загальний план проектів починає змінювати перспективу базуючись на положенні користувача. Гіпотеза полягає в тому що це зацікавить користувача.

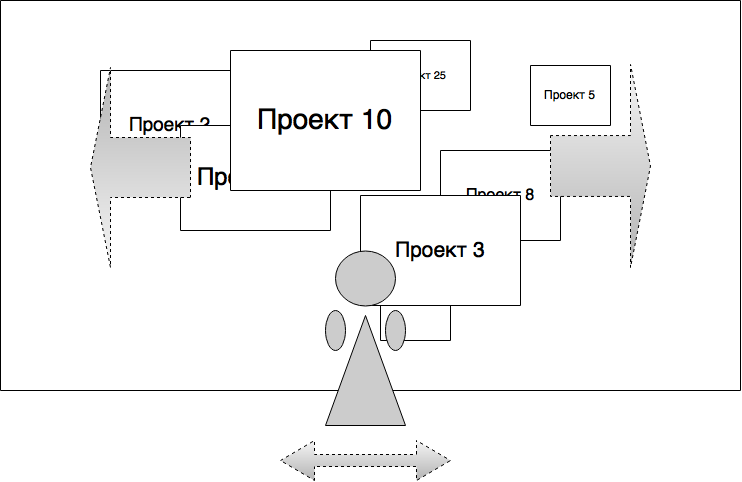


Рисунок 5.1 Відповідь інтферфейсу на рух користувача

Інтуїтивність – висока, від користувача не потребується жодних знань.

Конфлікти – відсутні для користувача, можуть бути конфлікти між користувачами

#### Зміна загального плану на режим перегляду проекту

На загальному плані проектів користувач може підійти ближче до екрану. Система переключиться в режим перегляду проекту. Якщо користувач відійде, то переключиться на загальний план.

Інтуїтивність – середня, практика показала що потрібно додаткова мотивація підійти.

Конфлікти – відсутні.

#### Жест горизонтального «перегортання» проектів

На екрані перегляду проекту користувач може «прегортати» сторінки з проектами. Рух правої руки з права на ліво, означає перегортання вліво, і навпаки.

Інтуїтивність – висока, деякі люди користувачі пробували використовувати жест «перетягування»

Конфлікти – повернення руки конфліктує з рухом в інший бік, також конфліктує з іншими жестами

#### Відповідь на рух користувача в режимі перегляду проекту

#### Жест протягування проектів для зміни проекту

#### Жест вертикального «перегортання» сторінок проекту для перегляду

#### Жест вертикального перетягування проекту для перегляду проекту

Загалі

# Вибір засобів реалізації

## Датчики жестів NUI

В цій секції проводиться аналіз існуючих датчикі жестів в рамках рішення. Головні показники для вибору:

Радіус дії датчика – рішення має працювати на відстані 1-5 метри.

Жести тіла – підтримка датчиком жестів не тільки рук, але і всього тіла.

Кількість об’єктів стеження – потенційно рішенням може користуватися більше ніж одна людина, це означає що датчик має підтримувати більше ніж одну людину

Спеціальне обладнання – датчик може потребувати додаткового обладнання

SDK – наявність бібліотек для розробки під потрібну платформу

Ціна – ціна датчика для інсталяції і розробки

Були розглянуті такі датчики:

Leap Motion – датчик ближньої дії, загальне застосування це керування комп’ютерними програми в звичайному режимі користуванні. Для нашого рішення не підходить, тому що потрібно розпізнавати жести на відстані від 1.5 метрів.

Microsoft Kinect 1.0 – передне покоління ігрових датчиків для Xbox.

Microsoft Kinect 2.0 – поточне покоління ігрових датчиків Xbox, адаптованих для промислового використання. Існують версії для PC.

PlayStation Move – ігровий датчик від Sony. Має кращу точність ніж Microsoft Kinect, але потребує додаткового датчика в руці користувача.

Таблиця 6.1Порівнянльна таблиця датчиків жестів

| Датчик | Опис | Радіус дії датчика(м) | Жести тіла | Кількість об’єктів стеження | Спеціальне обладнання | SDK | Ціна |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Leap Motion | Датчик жестів на короткій дистанції | 0.3-1 | Ні | 1 | Ні | .NET | 2000 грн |
| Microsoft Kinect 1.0 | Ігровий датчик | 2-6 | Так | 2 | Ні | .NET | 2600грн |
| Microsoft Kinect 2.0 | Ігровий та промисловий датчик | 0.3-6 | Так | 5 | Ні | .NET | 5000грн |
| PlayStation Move | Ігровий датчик | 2-6 | Ні | 1 | Так – датчик в руці | Ні | - |

Базуючись на потрібних характеристиках було обрано датчик Microsoft Kinect 2.0 для PC.

## Бібліотеки виявлення жестів

TBD

Kinect 2.0 SDK

OpenNI

OpenKinect

etc

## Платформа для реалізації презентаційного інтерфейсу

Презентаційний інтерфейс це головний інтерфейс який користувачі бачать і використовують. Для вибору платформи реалізації враховувались такі основні особливості.

Швидкість промальовування – інтерфейс активно використовує анімації, складні сцени з ефектами 3D і рендерить їх на великому екрані. Дуже важливо мати підтримку апаратного прискорення.

Наявність бібліотек UI – для проекту важливо мати підтримку широкого спектру бібліотек. В майбутньому планується розширення можливостей.

Комунікація з Kinect - основна маса інтерактивності інтерфейсу базується на реакції на події які генерує Kinect.

Найбільш поширеним способом для розробки додатків з багатим інтерфейсом, які можуть взаємодіяти з Kinect є використання Windows Presentation Foundation (WPF). WPF це дуже потужна бібліотека UI, що дозволяє створювати практично будь-який парадигми призначеного для користувача інтерфейсу. WPF використовує Direct3D, а отже є аппаратно прискоренною платформою.

Другий за поширеністю спосіб це використання HTML5/JavaScript. Сучасні браузери також використовують апаратне прискорення. На відміну від WPF не дозволяє працювати з Kinect на пряму.

Таблиця 6.2 Аналіз платформ для презентаційного інтерфейсу

|  |  |  |  |
| --- | --- | --- | --- |
| Вимога | WPF | HTML5/JS | Direct3D |
| Швидкість промальовування | Апаратно прискорене | Апаратно прискорене, але використовуючи DOM | Апаратно прискорене |
| Наявність бібліотек UI | Не відповідає потребам проекту | Дуже широке | Підтримує тільки ручний рендерінг |
| Комунікація з Kinect | Пряма | Через сервер (WebSocets) | N/A |
| Відкритий код | Ні | Частково | Ні |

Аналіз показав, що HTML/JS надає значну перевагу в наявності бібліотек, а також достатню швидкість промальовування. Таким чином, було прийнято рішення не використовувати WPF, і замінити його на HTML/JS, в якості базової бібліотеки інтерфейсу було обрано AngualrJS.

## Комунікація реального режиму часу між інтерфейсом користувача і обробником Kinect

Один з викликів цього рішення це комунікація реального часу між інтерфейсом користувача і сервером інсталяції. В режимі звичайної роботи, передаються такі типи даних:

Таблиця 6.3 Типи комунікації між інтерфейсом користувача і сервером інсталяції

|  |  |  |  |
| --- | --- | --- | --- |
|  | Напрямок | Об’єм даних | Опис |
| Виявлені жести | UI | Малий | Передається ключ виявленого жесту, і невеликий об’єм мета інформації. |
| Інтеракції | UI | Середній | Постійний потік даних з координатами важливих точок в просторі, положенням рук, тощо. |
| Відео | UI | Великий | Потік відео фреймів для тестових сценарієв |
| Команди | Сервер | Малий | Передається команда до серверу, така як оновити оцінки. |

Основні фактори які були прийняті до уваги:

Функціональність – обрана технологія має мати можливість реалізувати функціонал

Дуплексний режим – можливість роботи як від сервера до клієнта, так і навпаки

Швидкість – швидкість передачі даних, затримка відповіді, тощо

Для вибору було розглянуто три варіанти. HTTP – звичайні запити на сервер. HTTP Poll – довгі запити на сервер, які дозволяють передавати данні на клієнт без очікування з’єднання. WebSockets – відносно новий стандарт який привносить в веб двонаправлені сокети для комунікації з серевером.

Таблиця 6.4 Аналіз варіантів комунікації між інтерфесом користувача і сервером

|  |  |  |  |
| --- | --- | --- | --- |
|  | HTTP | HTTP Poll | WebSockets |
| Функціональність | Ні | Так, але потрібно вирішувати проблеми довгих запитів | Так, потребує функціонал перепідключення до сервера. |
| Дуплексний режим | Ні | Емулюється | Так |
| Швидкість | Висока, але не реального часу | Висока, але не реального часу | Дуже висока, за рахунок використання бінарних данних |
| Відео | Так, але потребує стрімінг | Так, але потребує стрімінг | Так |

Аналіз показав, що найкращим рішенням є WebScokets які мають швидкість достатню навіть для пофреймового передавання відео, мають мінімальну затримку за рахунок постійно відкритого з’єднання, а також доступні в усіх сучасних бразуерах.

## Комунікація в режимі підписки між сервером інсталяції і сервером статистики

TBD

# РеалізаціЯ

## Основні актори та use-case системи

Секція описує основних авторів і пов’язані з ними use-case.

Таблиця 7.1 Основні актори системи

|  |  |
| --- | --- |
| Користувачі | Сторонні люди які користуватимуться системою |
| Адміністратор | Персонал який налаштовує інсталяцію |

Користувач використовує NUI для своїх use-case, натомість адміністратор використовує звичайний веб інтерфейс. В нашій системі адміністратор це сукупна роль, яка робить і адміністрування і налаштування і звітність.

Користувач може бачити загальний план який в довільному порядку показує список проектів доступних для перегляду. Користувач може вибрати один з проектів, переглянути його, і оцінити його.

Адміністратор може подивитись звітність по оцінках і використанню.



Рисунок 7.1 Основні use-case системи

## Компоненти системи

Рішення виконане у вигляді поєднання програмного рішення і апаратних датчиків. Система складається з двох автономних пакетів:

Front-end – аплікація для користувачів рішення. Складається з інтерфейсу користувача, Kinect, сервера обробки даних з Kinect та файлової бази даних для реєстрації проектів. Розгортається на комп’ютері з підключеним Kinect.

Back-end – аплікація для адміністраторів рішення, а також для централізованого зберігання статистики. Складається з адміністративного інтерфейсу, серверу статистики і бази даних статистики.

На діаграмі нижче показані зв’язки між компонентами системи, а також актори системи.

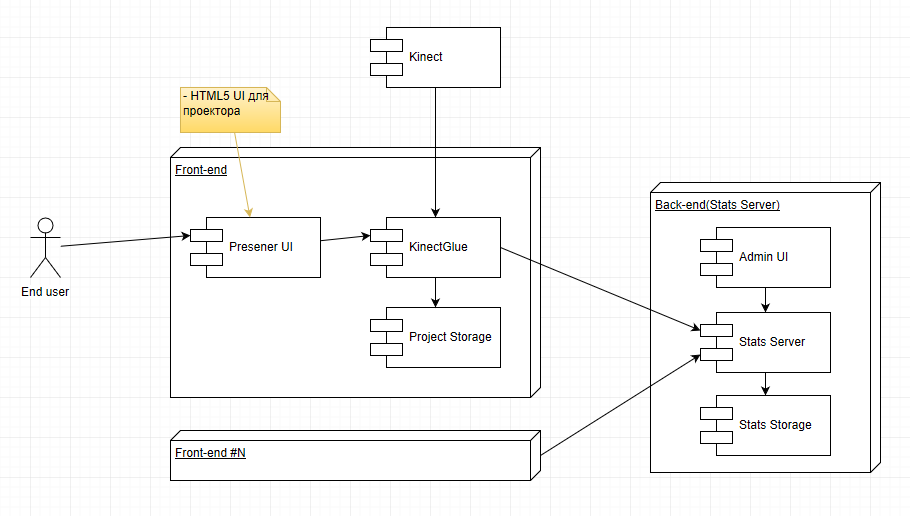


Рисунок 7.2 Компонентна діаграма рішення

Компоненти рішення працюють автономно і можуть складати інші топології. Наприклад інсталяцій Front-end може бути більше ніж одна, і вони будуть об’єднані в мережу розподілених інсталяцій, що матимуть спільну статистику і спільне адміністрування.

Модулі системи:

Presenter UI – Інтерфейс користувача, HTML5 інтерфейс який буде виводитись за допомогою проектора на великий екран.

KinectGlue – центральний сервер інсталяції, обробляє потік даних з Kinect: реалізовує алгоритми детектування жестів, оркестровці користувачів, тощо. Надає API для Presenter UI. Підключається в режимі підписки до сервера статистики і інформує Presenter UI з новою статистикою.

Project Storage – проста файлова БД в якій зберігається контент проекту його метаінформація.

Admin UI – інтерфейс адміністратора, дозволяє дивитись агреговану статистику, а також керувати проектами.

StatsServer – центральний сервер статистики, обробляє виклики з серверів інсталяції, а також підготовлює данні для звітів на Admin UI

StatsStorage – вбудована реляційна БД, зберігає агреговану статистику по проектах

## Взаємодія між компонентами

Ця секція описує комунікаційні протоколи між компонентами системи.

Комунікація між різними компонентами має різні вимоги, тому для них були використані різні технлогії.

Для комунікації між інтерфейсом користувача і сервером інсталяції використовується двонаправлені WebSocket. Для комунікації між сервером інсталяції і сервером статистики використовується схема Pub/Sub реалізована задопомогою in-memory шини ZeroMQ.

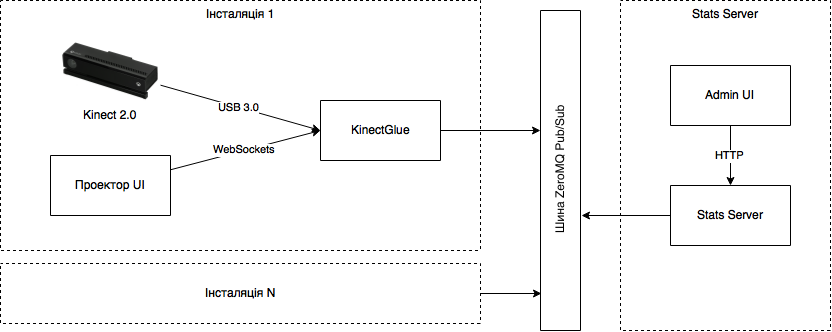


Рисунок 7.3Протоколи і взаємодія між компонентами

## Технічний стек

Ця секція описує технічний стек фінального рішення. Для прийняття рішення використовувались такі основні фактори:

Відповідність функціональним вимогам - можливіть сторити рішення в рамках заданих вимог

Ціна рішення – перевага була надана компонентам з відкритим кодом

Доступність знань – перевага була надана компонентам з якими був існуючий досвід

### Технічний стек Presenter UI

Інтерфейс користувача побудовано за допомогою веб технологій (TBD мотивацію), базуючись на бібліотеках з відкритим кодом.

Таблиця 7.2 Технічний стек Presenter UI

| Назва | Технологія | Мотивація |
| --- | --- | --- |
| Платформа | HTML5, JavaScript, CSS | * Наявність бібліотек * Можливість розділення логіки, представлення, та стилів |
| Бібліотека інтерфейсу | AngularJS | * Дозволяє писати складні веб сторінки |
| Бібліотека графіки | D3 | * Дозволяє візуалізовувати данні (в нашому випадку точки в просторі) |
| Юніт тести | Jasmine | * JavaScript UnitTest бібліотека |
| Менеджер пакетів | Bower | * Підходить для сайтів без компіляції |

### Технічний стек KinectGlue

Центральний сервер інсталяції побудовано використовуючи платформу .NET яка є ідеальною для роботи з Kinect SDK. Центральний сервер інсталяції використовує чотири типи комунікації. Внутрішню на основі React Extensions, зовнішню для UI – WebSockets і для комунікації з сервером статистики ZeroMQ, а також USB 3.0 для комунікації з Kinect.

Таблиця 7.3 Технічний стек KinectGlue

| Назва | Технологія | Мотивація |
| --- | --- | --- |
| Мова програмування | C# | * Чудова підтримка асинхронного програмування * Мова по замовчуванню для .NET платформи |
| Платформа | .NET | * Чудова підтримка Kinect SDK |
| Внутрішня комунікація | Reactive Extensions | * Реактивна, не блокуюча парадигма * Гнучке фільтрування та абстракція |
| Зовнішнє WebSocket АПІ | Fleck | * Підтримка протоколу WebSockets для передачі даних в реальному режимі часу |
| Юніт тести | NUnit | * .NET UnitTest бібліотека |
| Менеджер пакетів | Nuget | * Менеджер пакетів для .NET |

### Технічний стек Admin UI

Інтерфейс адміністратора побудовано за допомогою веб технологій, базуючись на бібліотеках з відкритим кодом.

Таблиця 7.4 Технічний стек Admin UI

| Назва | Технологія | Мотивація |
| --- | --- | --- |
| Платформа | HTML5, JavaScript, CSS | * Наявність бібліотек * Можливість розділення логіки, представлення, та стилів |
| Бібліотека інтерфейсу | ReactJS | * Дозволяє писати складні веб сторінки |
| Юніт тести | Jasmine | * JavaScript UnitTest бібліотека |
| Менеджер пакетів | npm | * Підходить для сайтів з компіляцією |

### Технічний стек StatsService

Сервер статистики побудовано використовуючи платформу .NET яка підходить для написання такого роду серверів. Сервер статистики використовує три типи комунікації. Внутрішню на основі React Extensions, зовнішню для UI – HTTP JSON і для комунікації з серверами інсталяцій ZeroMQ. В якості бази даних використовується внутрішня БД SQLite.

Таблиця 7.5 Технічний стек StatsServer

| Назва | Технологія | Мотивація |
| --- | --- | --- |
| Мова програмування | C# | * Чудова підтримка асинхронного програмування * Мова по замовчуванню для .NET платформи |
| Платформа | .NET | * Добре підходить для таких рішень |
| Внутрішня комунікація | Reactive Extensions | * Реактивна, не блокуюча парадигма * Гнучке фільтрування та абстракція |
| Зовнішнє HTTP JSON API | NancyFx | * Проста бібліотека для написання HTTP JSON API |
| Юніт тести | NUnit | * .NET UnitTest бібліотека |
| Менеджер пакетів | Nuget | * Менеджер пакетів для .NET |
| База даних | SQLite | * Проста в використанні реляційна БД * Поставляється у вигляді бібліотеки |

## Внутрішня організація системи

### Реактивна комунікація між компонентами серверу інсталяції (KinectGlue)

Система використовує шаблони реактивного програмування. Це обумовлено вимогами швидкої обробки даних, мінімальною затримкою (адже кожна затримка буде відбуватися на досвіді користувача). Вся система використовує неблокуючі реалізації.

Для обробки потоку подій в реактивному вигляді використовується .NET Reactive Extensions.

Схема потоку повідомлень серверу інсталяцій показана на діаграмі нижче.

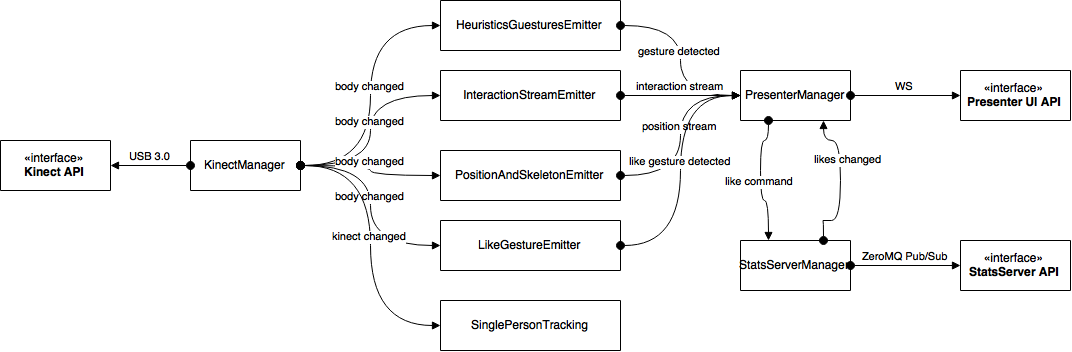


Рисунок 7.4 Схема потоку повідомлень сервера інсталяції

Найбільша частина повідомлень починає свій шлях від Kinect, зазвичай це інформація про положення скелету. Після обробки цієї інформації генеруються події з жестами, інтеракціями, тощо. Ці події спрямовуються на інтерфейс користувача чи сервер статистики. Між сервером статистики та інтерфейсом користувача також є обмін повідомленнями.

### Організація реалізації емітерів, виявлення рухів

Одна з головних частин системи це обробка потоку даних з Kinect, виявлення жестів, рухів. Обробка потоку виконана у вигляді емітерів, об’єктів які на вхід приймають потік фреймів, або поточний фрейм і базуючись на внутрішніх алгоритмах видають чи не видають результат.

Система емітерів показана на діаграмі нижче.

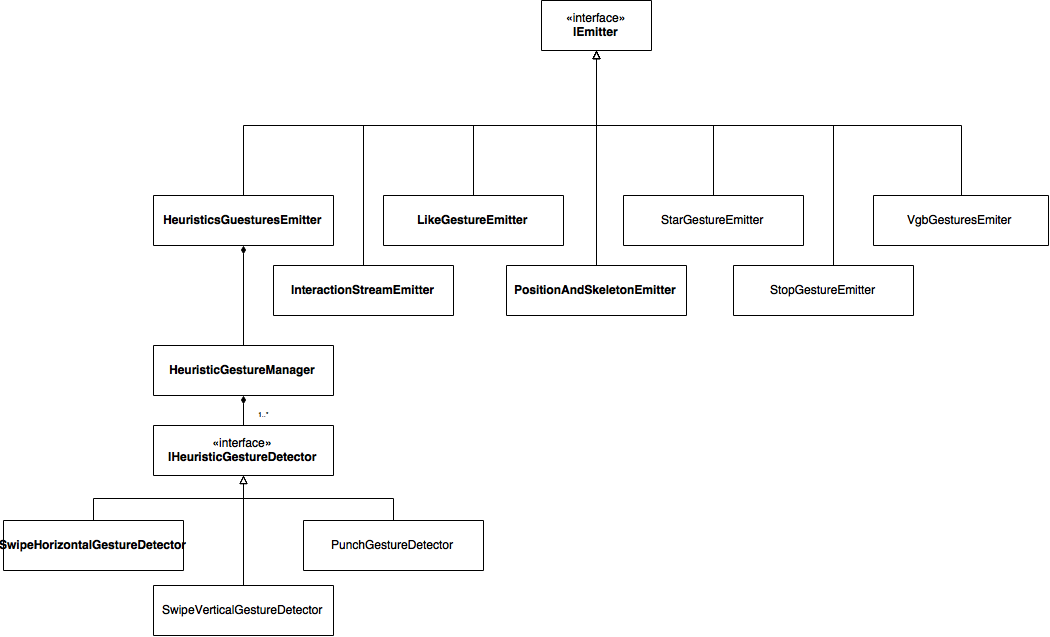


Рисунок 7.5Схема емітерів для обробки потоку даних

Всі емітери реалізовують маркуючий інтерфейс IEmitter. За рахунок відсутності загального контракту IEmitter не нав’язує свій контракт, а отже сторона що використовує конкретний емітер має використовувати конкретну реалізацію емітера.

HeuristicsGuesturesEmiter – емітер для евристичного виявлення жестів. Реалізований у вигляді набору детекорів.

SwipeHorizontalGestureDetector – виявляє горизонтальний жест перегортання сторінки. Використовується в фінальному рішенні.

SwipeVerticalGestureDetector – виявляє вертикальний жест перегортання сторінки. Не використовується в фінальному рішенні, практика показала гіршу інтуїтивність ніж жест захвату і протягування екрана.

PunchGestureDetector – виявляє жест схожий на удар в бік презентації, гіпотеза про те що це може використовуватись для оцінки проекту. Не використовується в фінальному рішенні, практика показала не інтуїтивність цього жесту, а також конфлікти з іншими жестами.

InteractionStreamEmitter – емітер стірму інтеракції. Фактично потік позицій правої руки і її стан. Використовується в фінальному рішенні для реалізація протягування презентації для прегляду.

LikeGestureEmitter – емітер жесту оцінки – оби дві руки над головою. Використовується в фінальній реалізації.

PositionAndSkeletonEmitter – емітер позиції та скелетона. Використовується в фінальній реалізації для відстеження положення користувача, а також на тестових екранах.

StarGestureEmitter – другий варіант емітеру для жеста оцінки. В цьому емітері також використовується інформація про положення ніг. Не використовується в фінальному рішенні, практика показала не інтуїтивність.

StopGestureEmitter – жест складених навхрест рук який мав використовуватись для відміни «активності» користувача, і автоматичної передачі іншому. Не використовується в фінальному рішенні, конфліктує з іншими жестами.

VgbGesturesEmiter – емітер жестів що будується на бібліотеці машинного навчання що входить в поставку Kinect SDK. Не використовується в фінальному рішенні, практика показала ненадійність виявлення жестів «перегоратння».

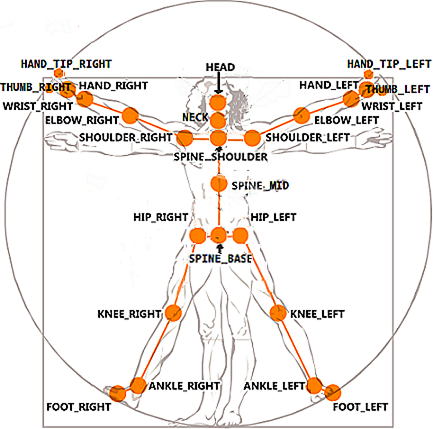
## Алгоритми виявлення жестів

### Короткі загальні відомості

Для виявлення жестів дуже важливо правильно обрати алгоритми, і данні з якими алгоритми будуть працювати. Kinect SDK 2.0 представляє декілька рівнів доступу.

RAW – прямий доступ до даних сенсору. Надаються аудіо, відео високої якості, а також один чи два канали інфрачервоного зображення високої якості. На цьому рівні немає інформації з якою було б легко працювати для виявлення жестів, проте наприклад відео інформацію можна використовувати на час розробки.

Скелетон – на цьому рівні SDK надає можливості отримати вже оброблений скелет людини чи людей за якими слідкує Kinect. Скелет надається у вигляді набору точок в просторі, а також з’єднань між ними. На цьому рівні можна розробляти свої алгоритми, що базуються на цих точках, а також часі.



Інтеракції – наступний рівень на якому SDK надає інформацію про інтерактивні рухи загалом. Інформація про рух складається з поточної позиції керуючої частини тіла (наприклад права рука), стан руки (наприклад: відкрита, захоплена) і стану перетягування.

Виявлення жестів - на цьому рівні SDK дає інструменти для того щоб записати рухи, відмаркувати в спеціальному редакторі і потім згенерувати алгоритм виявлення руху.

## Виявлення активного користувача

Активний користувач - користувач з яким в даний момент працює система. Датчик може стежити за 6 користувачами, користувачі можуть відходити і підходити. Також користувачі можуть «передавати» активність іншому користувачу.

Алгоритм пошуку активного користувача показано нижче на рис 7.2. Алгоритм запускається циклічно, паралельно до основної роботи. Данні про список тіл, та чи слідкує датчик за тілом отримується з датчику. Активний користувач зберігається в пам’яті програми.

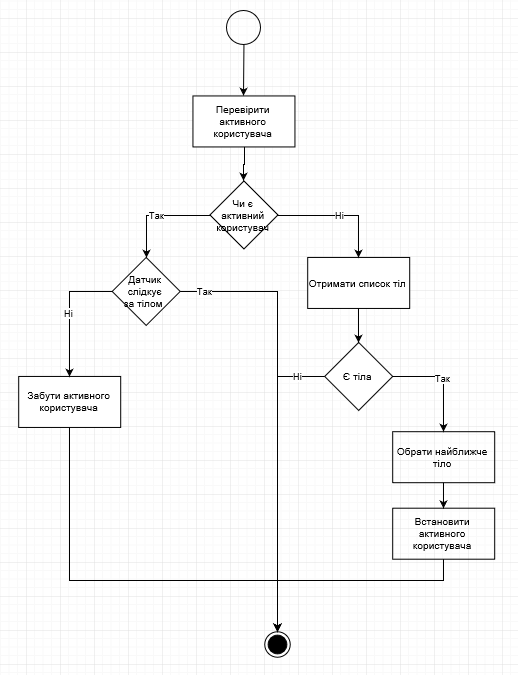


Рисунок 7.6 Алгоритм пошуку активного користувача

## Загальне бачення евристичних алгоритмів

В рішенні використовують різні типи алгоритмів. Проміж інших, виділяється група евристичних алгоритмів, які є схожими за поведінкою, а тому мають спільну основу. Усі ці алгоритми працюють з чергою точок у просторі у «вікні» часу. Результатом роботи алгоритму є або пропуск фрейму, або нотифікація виявленого руху. Нижче надано загальну схему алгоритму.

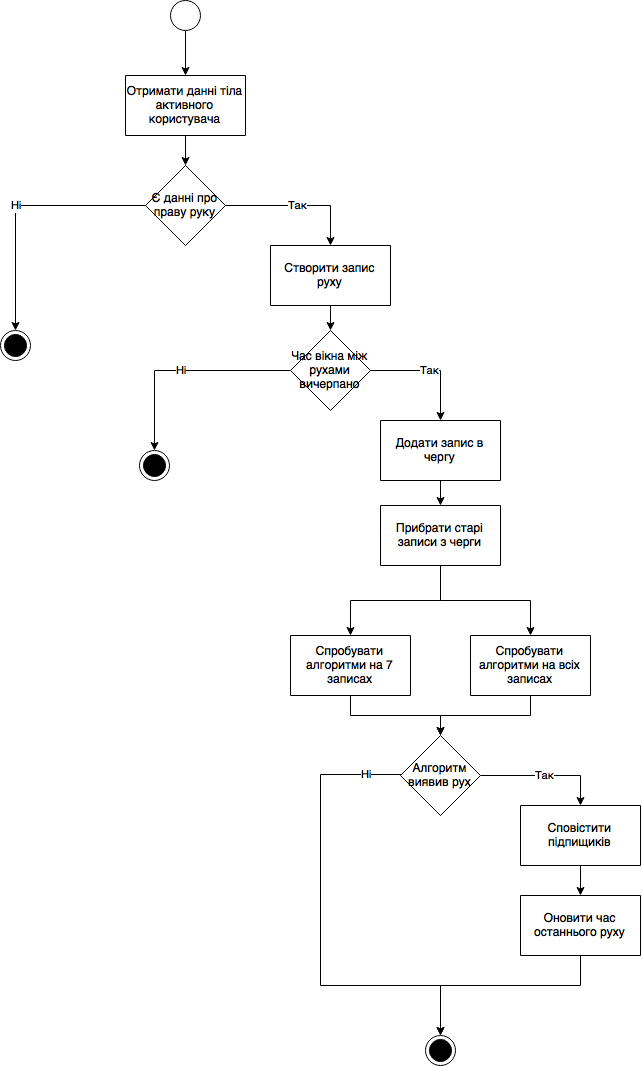


Рисунок 7.7 Загальна схема виконання я евристичних алгоритмів

## Виявлення руху «перегортання»

Для вибору конкретного проекту використовується рух «перегортання». Це горизонтальний рух з права на ліво – для перегортання вліво, і з ліва на права – в інший бік.

В рішенні реалізовано два варіанти, перший (SwipeHorizontalGestureDetector) використовує практики евристичного програмування, і використовується по замовчанню. Другий (VgbGesturesEmiter) використовує вбудований в SDK інструментарій для генерації детектора жестів. Для навчання алгоритму було записано рухи 12 людей. За рахунок особливостей руху, перший алгоритм дає кращі результати.

В цій секції описано перший алгоритм з евристичним виявленням руху, загальна схема виконання евристичних алгоритмів надана в секції 7.5.

Алгоритм виконується циклічно, і викликається паралельно з іншими евристичними алгоритмами. Алгоритм використовує експериментально підібрані коефіцієнти. Зазвичай коефіцієнти залежать від середовища в якому буде розгорнута інсталяція, і має підбиратись в кожному конкретному випадку.

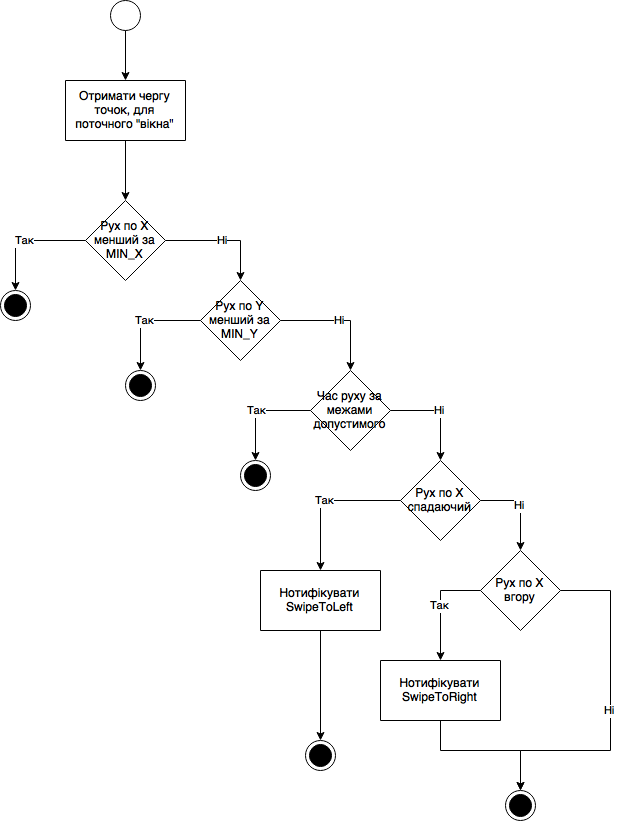


Рисунок 7.8 Алгоритм розпізнавання горизонтального руху «перегортання»

## Концепція фізичної інсталяції

Фінальне рішення складається з сенсора Kinect під’єднаного до сервера інсталяції, на якому йде обробка рухів користувачів, та інтерфейсу що реагує на рухи. За допомогою проектора інтерфейс користувача продукується на стіну або на великий екран. Приблизне розташування елементів інсталяції показано на діаграмі нижче.

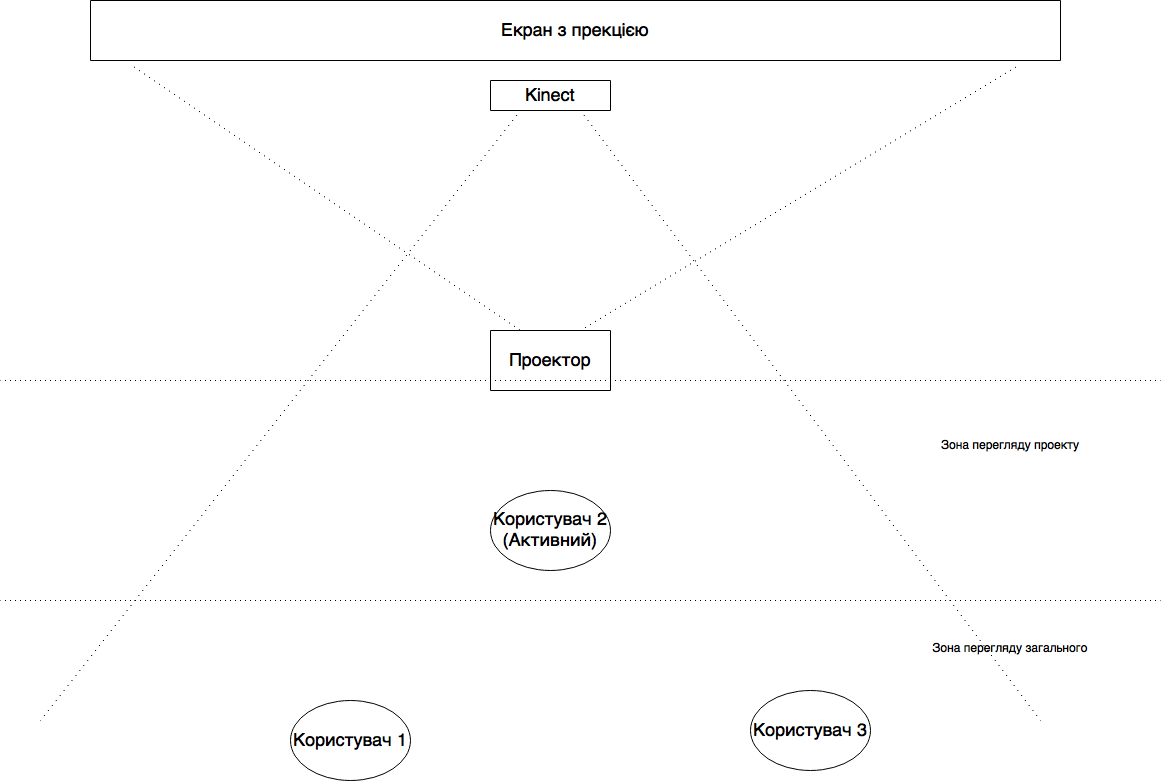


Рисунок 7.9Фізичне розташування елементів інсталяції

Найкращі результати дає робота з дуже великою проекцією, відчуття було протестоване на екрані 3 на 4 метри. Датчик руху Kinect має бути розташований на висоті приблизно 30-50см від поверхні підлоги.

## Результати роботи системи

### Інтерфейс користувача

Головний екран інтерфейсу користувача, показує проекти розміщуючи їх в тривимірному просторі, проекти з найбільшою кількістю оцінок – ближче, з меншою – далі, а також більш розмитої.

Для заохочення користувача в залежності від позиції відносно датчика руху, екран зміню точку зору, створюючи ефект паралакса.

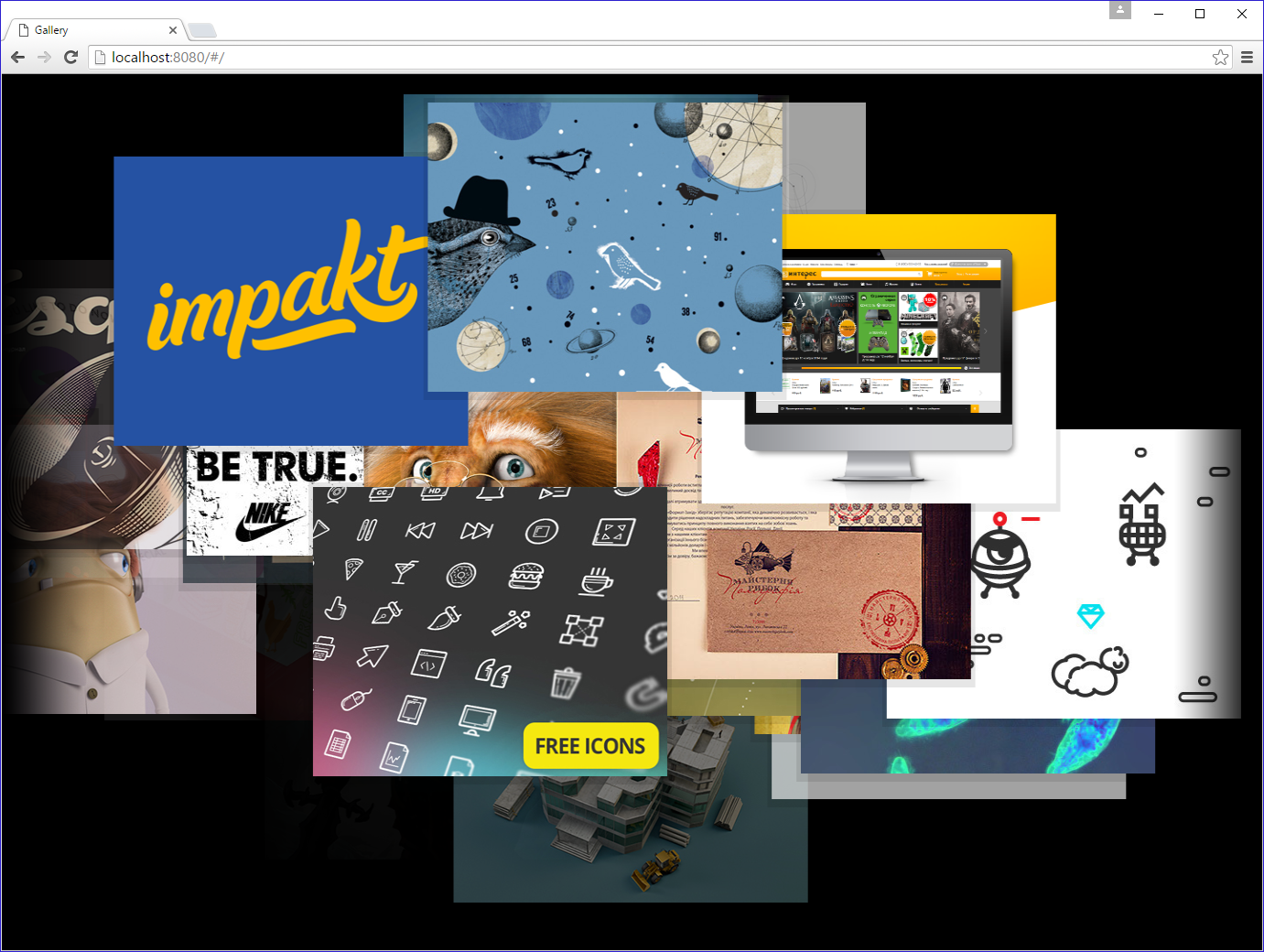


Рисунок 7.10Головний екран інтерфейсу користувача

Після того як користувач підходить ближче до датчика (виявлено що користувач зацікавився) автоматично змінюється на показ проектів. В цьому режимі користувач може «прогортати» вправо чи вліво для вибору проекту. Може «потягнути» зображення вверх чи вниз для того шо б переглянути проект.

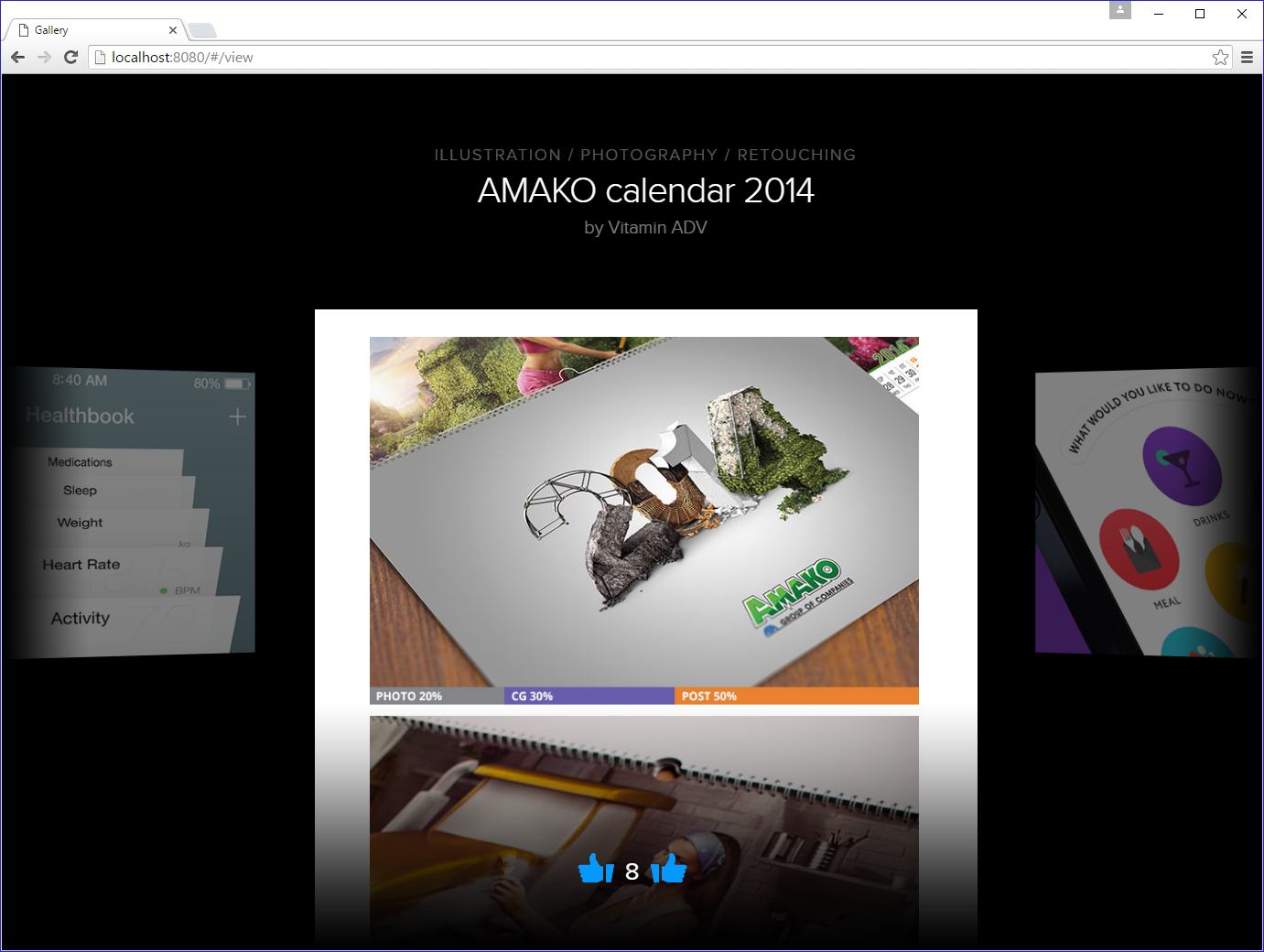


Рисунок 7.11 Екран інтферфейса користувача для перегляду проекта

Користувач має можливість оцінити проект. Для цього потрібно щоб датчик виявив жест «зірочка». Оцінка автоматично оновиться на всіх інсталяціях.

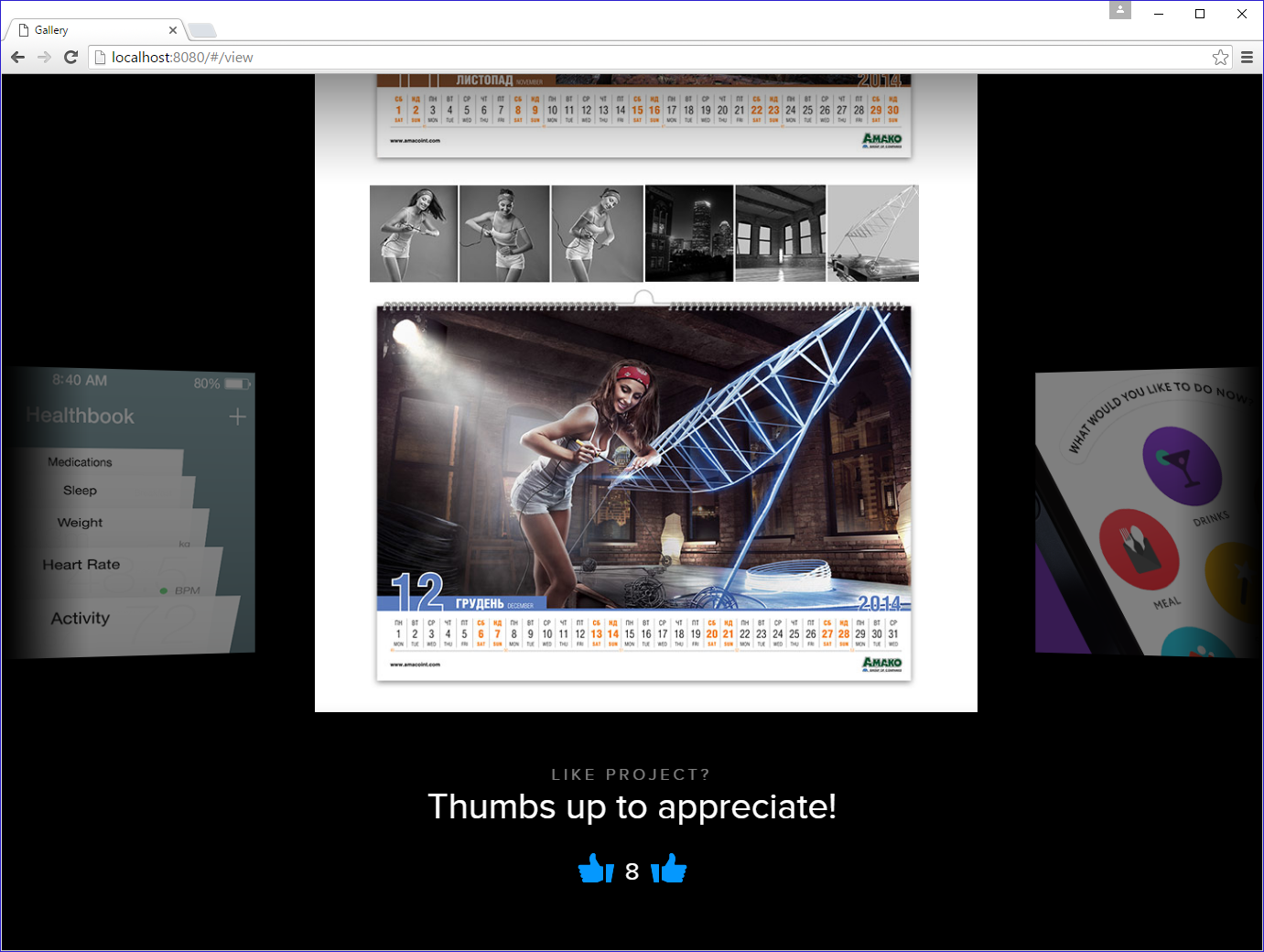


Рисунок 7.12 Екран оцінки проекта після того як було виявлено жест "зірочка"

Інтерфейс користувача дуже активно використовує анімацію для заохочення користувачів. В межах документа немає можливості показати анімацію.

### Інтерфейс адміністратора

На відміну від інтерфейсу користувача, адміністративний інтерфейс розрахований тільки для внутрішнього користування . Адміністративний інтерфейс надає можливість звітності.

# Висновки

# Список використаних джерел

Computer and Machine Vision, Fourth Edition: Theory, Algorithms, Practicalities

Practical Python and OpenCV + Case Studies

<https://en.wikipedia.org/wiki/Natural_user_interface>

<https://en.wikipedia.org/wiki/Technology_adoption_life_cycle>

# Додатки