Module 5 Journal

Hai Nguyen

CS 499 Computer Science Capstone

Professor Brook Goggin

21 Nov 2023

## Part One:

## Nano Structures:

Nanostructures encompass a variety of engineered structures, including nanotextured surfaces, nanoparticles, nanotubes, and more complex nano-scale structures. The nanoscale is defined as between 1 and 100 nanometers, making nanostructures the smallest possible structures, larger only than simple molecules or atoms. Nanostructures can be formed through top-down bulk processes, involving the successive breaking down of particles into nanoparticles. Nanostructures have diverse applications, including in materials with enhanced energy storage or catalytic performance, super-lenses, solar panels, catalysts, electrodes, hydrogen storage, solar thermal energy, and filtration.(Heney, 2022).

Nanostructures enable the development of smaller and more efficient electronic components. This could lead to the creation of smaller, faster, and more powerful computing devices. Nanoscale transistors and other components could contribute to advancements in processing speed and computational capabilities. Nanostructures can also contribute to the development of high-density, non-volatile memory storage devices. This could lead to improvements in data storage and retrieval, affecting the performance and efficiency of computer systems. Moreover, nanostructures can be used in the development of highly sensitive sensors and imaging devices. This may result in improved capabilities for medical diagnostics, environmental monitoring, and other applications, providing consumers and workers with more advanced tools for data collection and analysis (Heney, 2022). In summary, the integration of nanostructures into computing technologies has the potential to transform the landscape of computer science and, consequently, impact the experiences of individuals in various roles and contexts.

To prepare for a career involving nanostructure technology in computer science, I should consider taking relevant courses in materials science, semiconductor physics, nanotechnology, and quantum computing. Additionally, seeking internships or research opportunities in labs or companies working on nanostructure-related projects can provide hands-on experience and enhance my skill set in this exciting and rapidly evolving field.

As a student in the Computer Science program, my progression from the initial assignment into the program to the current stage reflects a dynamic evolution of my career plans and personal development. During the program’s early phase, I immersed myself in foundation courses (CS-200- Perspectives in the Social Science, IT-201-Computer Platform Technologies, IT-145-Foundation in Application Development, etc.), gaining exposure to diverse aspects of computer science. This period involved the exploration of distinct interests within the field, such as software development (CS-360), data science (DAD-220), data structures and algorithms (CS-260), and software reverse engineering (CS-410). At the beginning of the program, the student might have had a broad interest in computer science without specific career goals. The decision to enter the program could have been influenced by a curiosity about technology, programming, or a desire for a career in a dynamic and growing field.

## 3D Scanning Technology:

3D scanning technology is being employed by archaeologists globally to digitally preserve historical and cultural relics. The touch-free nature of 3D scanning allows for high-fidelity digital preservation without the risk of damage to delicate objects. 3D scanning also enables the creation of accurate and detailed digital twins of artifacts, serving as a permanent record that can be distributed and examined by experts worldwide. 3D scanning is presented as a more efficient and less risky alternative to traditional methods like photogrammetry, x-rays, or CT scans (Vakulenko, 2023).

The use of 3D scanning contributes to advancements in computer graphics and 3D modeling. Computer scientists can leverage these technologies to create more realistic and detailed virtual environments, which can be applied in fields such as gaming, virtual reality, and simulation. This can enhance immersive experiences for consumers and professionals alike. Moreover, 3D scanning technology plays a vital role in creating accurate 3D models of real-world objects and environments. This can be incorporated into augmented reality (AR) and virtual reality (VR) applications. Consumers may experience more realistic and interactive AR and VR content, whether it's for gaming, education, or virtual tourism. Overall, as technology evolves, it has the potential to shape various aspects of computer science and significantly influence the experiences of consumers, workers, and students in diverse fields.

I believe that 3D modeling, and 3D scanning technology offers multiple practical applications. I could work on developing algorithms and software tools for processing and rendering 3D scans, contributing to the creation of realistic virtual environments, characters, or objects. In addition, 3D scanning is crucial for creating realistic assets in virtual and augmented reality applications. If I was intrigued by VR/AR technologies, my skills could be applied to designing and optimizing immersive experiences. This may involve developing algorithms for integrating 3D scans into virtual environments or creating interactive AR applications. I also need to stay updated on the latest developments in 3D scanning and related fields that will be important for a dynamic and evolving career.

## Part Two:

### Status Checkpoints Artifact 1

* In the ClickedItemActivity.java, I aim to extend the functionality of the app by implementing innovative features or functionalities. This may involve using advanced mobile programming techniques, frameworks, or libraries to enhance the user experience or provide additional value to users. (Course Outcome 3)
* Practical Enhancements for ClickedItemActivity.java:
* Implement a feature that allows users to interact with the displayed item, such as adding comments or reviews.
* Enhance the user interface with more dynamic and interactive elements, like animations or gestures.
* Provide options for users to customize the displayed item's details or appearance based on their preferences. In this case, users can choose to display item’s text size (small, medium, or large).
* Original code:

ClickedItemActivity.java:

package com.zybooks.projecttwohainguyenui;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;

import android.os.Bundle;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.TextView;

public class ClickedItemActivity extends AppCompatActivity {

ImageView imageView;

TextView textView;

EditText editText;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_clicked\_item);

imageView = findViewById(R.id.imageView);

textView = findViewById(R.id.tvName);

editText = findViewById(R.id.inputText);

Intent intent = getIntent();

if (intent.getExtras() != null) {

String selectedName = intent.getStringExtra("name");

int selectedImage = intent.getIntExtra("image", 0);

textView.setText(selectedName);

imageView.setImageResource(selectedImage);

}

}

}

Modified code with 3 enhancements: ClickedItemActivityModified.java.

With 3 modifications, the ClickedItemActivityModified will now allow users to add comments, provide dynamic animations, and potentially customize the displayed item's details.

“package com.zybooks.projecttwohainguyenui;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.Spinner;

import android.widget.TextView;

public class ClickedItemActivity extends AppCompatActivity {

ImageView imageView;

TextView textView;

EditText editText;

Spinner textSizeSpinner; // Added Spinner for text size selection

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_clicked\_item);

imageView = findViewById(R.id.imageView);

textView = findViewById(R.id.tvName);

editText = findViewById(R.id.inputText);

textSizeSpinner = findViewById(R.id.textSizeSpinner); // Initialize spinner from layout

Intent intent = getIntent();

if (intent.getExtras() != null) {

String selectedName = intent.getStringExtra("name");

int selectedImage = intent.getIntExtra("image", 0);

textView.setText(selectedName);

imageView.setImageResource(selectedImage);

// Enhancement 1: Allowing users to add comments or reviews

editText.setOnFocusChangeListener(new View.OnFocusChangeListener() {

@Override

public void onFocusChange(View view, boolean hasFocus) {

if (!hasFocus) {

String userComment = editText.getText().toString();

// Save the comment to data structure or send it to a server

}

}

});

// Enhancement 2: Adding animations such as fade in to the ImageView

imageView.animate().alpha(1.0f).setDuration(1000);

// Enhancement 3: Allowing users to customize the displayed item's details

// Provide options to change the text size based on user preferences.

// Set up the spinner with predefined text size options

ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(

this,

R.array.text\_size\_options,

android.R.layout.simple\_spinner\_item

);

adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);

textSizeSpinner.setAdapter(adapter);

// Handle spinner item selection to change text size

textSizeSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

@Override

public void onItemSelected(AdapterView<?> adapterView, View view, int position, long id) {

// Get the selected text size from the spinner

String selectedTextSize = adapterView.getItemAtPosition(position).toString();

// Set the text size based on the selected option

switch (selectedTextSize) {

case "Small":

textView.setTextSize(14);

break;

case "Medium":

textView.setTextSize(18);

break;

case "Large":

textView.setTextSize(24);

break;

}

}

@Override

public void onNothingSelected(AdapterView<?> adapterView) {

// Do nothing if nothing is selected

}

});

} } }”

|  |  |  |
| --- | --- | --- |
| 1 | Artifact 1: Software Design & Engineering  Weigh Tracking App |  |
| 2 | Working on Initial Enhancements |  |
| 3 | Submitted, Awaiting Instructor Feedback |  |
| 4 | Working on Final Enhancements |  |
| 5 | Upload to ePortfolio |  |
| 6 | Finalized ePortfolio |  |

### Status Checkpoints Artifact 2

* Practical Enhancement: Further optimization and evaluation of sorting algorithms, particularly quicksort, to improve their efficiency and performance.
* Course Outcome Alignment: This enhancement aligns with Course Outcome 1, as it aims to demonstrate proficiency in sorting algorithms and their optimization, which can support diverse audiences in making informed decisions by enhancing computer system and application performance.
* Original code:

void quickSort(vector<Bid>& bids, int begin, int end) {

int mid = 0;

//if zero or one bid to sort, then done

if (begin >=end) {

return;

}

// partition bids into low and high parts

mid = partition(bids, begin, end);

//recursively call quicksort using midpoint value (begin to end)

quickSort(bids, begin, end);

//recursively call quicksort using midpoint value (mid + 1, end)

quickSort(bids, mid + 1, end);

}

* Modified code: Switch to a different sorting algorithm, insertion sort when the partition size becomes small (< 10). The idea is that insertion sort can be more efficient than quicksort for small datasets due to its lower overhead. Overall, this optimization aims to reduce the overhead of recursive calls in quicksort for small partitions, contributing to better performance in scenarios where quicksort may be less efficient due to the associated function call overhead.

void quickSort(vector<Bid>& bids, int begin, int end) {

int mid = 0;

// Optimization: Use insertion sort for small partitions

if (end - begin + 1 <= 10) {

// Insertion sort implementation

for (int i = begin + 1; i <= end; ++i) {

Bid key = bids[i];

int j = i - 1;

// Move elements of bids[begin..i-1] that are greater than key.title

// to one position ahead of their current position

while (j >= begin && bids[j].title.compare(key.title) > 0) {

bids[j + 1] = bids[j];

--j;

}

// Insert the key into the appropriate position

bids[j + 1] = key;

}

} else {

// Continue with quicksort for larger partitions

mid = partition(bids, begin, end);

quickSort(bids, begin, end);

quickSort(bids, mid + 1, end);

}

}

|  |  |  |
| --- | --- | --- |
| 1 | Artifact 2: Data Structures and Algorithms  Sorting Algorithms in VectorSorting.cpp |  |
| 2 | Working on Initial Enhancements |  |
| 3 | Submitted, Awaiting Instructor Feedback |  |
| 4 | Working on Final Enhancements |  |
| 5 | Upload to ePortfolio |  |
| 6 | Finalized ePortfolio |  |

### Status Checkpoints Artifact 3

* For the ProjectTwoDashboard.ipynb artifact, I can align practical enhancements and intended course outcome 3: Building a full-stack application using different programming languages (e.g., Node.js for the back-end), demonstrate the ability to work with multiple programming languages to create a full-stack application.
* Implement a backend using Node.js, Express, and MongoDB to handle data storage and retrieval.
* Create RESTful API endpoints in Node.js to communicate with MongoDB, allowing the Dash application to fetch and update data.
* Adjust the current MongoDB operations in your Python code to utilize the API endpoints provided by your Node.js backend.
* Ensure that the Dash application interacts with MongoDB through your Node.js server rather than directly.
* Set up communication between your Dash application and the Node.js backend. Dash can make HTTP requests to the Node.js server to fetch data or send updates.
* This outcome aligns well with the development of a data dashboard using Dash, which involves integrating Python (Dash) for the front-end with MongoDB as a NoSQL database on the back-end.

|  |  |  |
| --- | --- | --- |
| 1 | Artifact 3: Databases  International Animal Shelter Project  ProjectTwoDashboard.ipynb |  |
| 2 | Working on Initial Enhancements |  |
| 3 | Submitted, Awaiting Instructor Feedback |  |
| 4 | Working on Final Enhancements |  |
| 5 | Upload to ePortfolio |  |
| 6 | Finalized ePortfolio |  |