Hermann-von-Helmholtz-Gymnasium Potsdam

Abschlussprojekt Informatik

Snake II

Ein Spiel basierend auf dem Klassiker von Nokia, erstellt mit Lazarus

Name: Carl Hauschke und Konrad Lange

Lehrer: Frau Handke

Datum: 17.03.2017

Gliederung

1. Thema und Aufgabenstellung

Jeder, der früher ein Handy von Nokia besessen hat, kennt bestimmt das Spiel „Snake“. In dem Spiel muss der Spieler, mit Hilfe der Tasten, eine Schlange so steuern, dass zum einen möglichst viel Futter gesammelt wird und zum anderen Kollisionen mit dem Schwanz vermieden werden. Dieser wird mit dem Einsammeln von Futter immer länger, was das Spiel erschwert. Die Schlange kann nach oben, unten, links und rechts gesteuert werden, wobei Kopf zuerst die neue Richtung aufnimmt und die restlichen Elemente des Schwanzes jeweils die Position ihres Vorgängers einnehmen. Das Erreichen der Bildränder stellt keine Kollision dar, die Schlange erscheint wieder auf der jeweils gegenüberliegenden Seite des Bildes. Das Spiel ist beendet, wenn der Spieler den Kopf der Schlange gegen eines der Schwanzelemente steuert. Dieses Spiel möchten wir mit Lazarus umsetzten. Dabei wollen wir zusätzlich in einer externen Datei die Höchstpunktzahl speichern.

2. Darlegung der Lösungsstrategie

Zuerst möchten wir die Abfrage der Pfeiltasten realisieren, um den Kopf der Schlang steuern zu können. Dieser soll, wie auch jedes Schwanzelement der Schlange und das Futter ein Bild sein. Die Bilder erstellen wir selbst mit Photoshop. Jedes dieser Bilder hat auf dem Spielfeld eine X und eine Y Koordinate. Wenn die Koordinaten des Schlangenkopfes mit den Koordinaten des zufällig auf dem Spielfeld erstellten Futters übereinstimmen, wird die Schlange um ein Schwanzelement erweitert. Außerdem wird der Punktestand um eins erhöht, das Futter verschwindet und wird an einer neuen zufälligen Stelle neu erstellt. Die zweite Art von Kollision kommt zustande, wenn die Koordinaten des Kopfes mit den Koordinaten eines Schlangenelements übereinstimmen. Dabei kommt es zum Abbruch des Spiels und der Spielstand wird angezeigt. Den Highscore möchten wir in einer externen Datei speichern, damit sich die Spieler messen können.
Bei der Bewegung der Schlange nimmt jeweils das hintere Schwanzelement die alten Koordinaten des Vorgängers, beziehungsweise das erste Schlangenelement die alten Koordinaten des Schlangenkopfes an. Wenn die Schlange das Ende des Fensters erreicht, soll sie auf der gegenüberliegenden Seite wieder in das Bild bewegt werden. Das Erstellen und anhängen neuer Schwanzelemente könnte eine der schwersten Teilaufgaben sein, da die Anzahl der Schwanzelemente nicht vorher festgelegt werden kann und wir sie deshalb dynamisch bei Ausführung des Programms erstellen müssen.

3. Realisierung der Lösungsstrategie

Bei der Realisierung unseres Spiels hatten wir folgenden Ablauf. Zuerst haben wir mit der Umsetzung des Kopfes angefangen, der sich mit den Pfeiltasten bedienen lässt und bei Erreichen des Bildschirmrandes auf die gegenüberliegende Seite teleportiert wird. Dazu mussten wir erst einmal einen Weg finden, um die Pfeiltasten abzufragen, bevor wir die Koordinaten des Kopfes in die jeweilige Richtung um eine Konstante Delta verschieben konnten. Nachdem der Kopf sich frei auf dem Spielfeld bewegen ließ, widmeten wir uns dem Essen, welches mit Beginn des Spiels zufällig auf dem Bildschirm erstellt wird und bei Kollision mit dem Kopf neue zufällige Koordinaten zugewiesen bekommt, wobei der Score erhöht wird.
Nachdem der Score bereits gezählt werden konnte, war der nächste Schritt, ein Hauptmenü und eine Anzeige zu erstellen. Diese zeigt während des Spiels den aktuellen Score an, wobei dies dazu führte, dass die Teleportation des Kopfes wieder verändert werden musste, da sonst eine Überlappung mit der Anzeige des Scores möglich gewesen wäre.
Um nun ein funktionsfähiges Spiel zu erhalten, war der nächste Schritt, die Schlange zu entwickeln, die an den Kopf angehängt wird und nach jedem Verzehr eines Futters länger wird. Der Kopf musste nun noch um die Möglichkeit erweitert werden, eine Kollision mit dem Schwanz festzustellen, um das Spiel zu beenden.
Nachdem das Spiel funktionsfähig war, wurde der letzte optische Feinschliff vorgenommen und das Speichern des höchsten erreichten Scores in einer externen Datei auf dem Computer eingebaut und damit dieser im Hauptmenü des Spiels angezeigt werden kann.

<u>Beschreibung der Funktion Schwanzerstellen</u>

In dieser Funktion wird der Schlange ein neues Element angefügt.
Realisiert wird dies, indem Bildobjekte in einem Feld gespeichert werden, welches sich dynamisch vergrößert. Bei jedem Aufrufen der Funktion wird das Feld um einen weiteren Platz ergänzt, und anschließend wird in diesem ein Bild erstellt, welches dann bearbeitet wird. Das Bild, welches angezeigt wird, wird geladen und auf die richtige Größe gebracht.
Bei jedem Aktualisieren der Koordinaten für die Schlange werden die alten Koordinaten der einzelnen Elemente in einem anderen Feld abgespeichert. Mit Hilfe dieser Koordinaten kann nun die Position des neuen Elements bestimmt werden.

Reflektierend war der größte Fehler in unserer Strategie, dass wir uns am Anfang zu sehr auf die graphische Umsetzung des Problems und User Interface fokussiert haben und es logischer und vorteilhafter gewesen wäre zuerst die technische Umsetzung und Anwendung umzusetzen. Das ist der Punkt der uns während der Realisierung besonders aufgefallen ist.

4. Selbsteinschätzung

Im Allgemeinen konnten wir unser Arbeitsziel erfolgreich umsetzen. Das Spiel erfüllt alle Punkte, die wir uns als Ziel gesetzt hatten, jedoch könnte man das Spiel noch um weitere Spielmodi erweitern, wie beispielsweise erhöhte Schwierigkeitsgrade, bei denen die Schlange schneller ist oder in kürzerer Zeit stärker an Länge zunimmt. Auch können wir uns einen Multiplayer-Modus vorstellen, bei dem mit zwei Schlangen gleichzeitig gegeneinander gespielt werden kann. Technisch könnte noch ein in der Größe veränderbares Spielfeld umgesetzt werden und mit passenden Tönen zum Spiel könnte man die Atmosphäre noch verstärken. Außerdem gibt es bei der Effizienz des Programms ein Verbesserungspotential, so denken wir zum Beispiel, das eins der Arrays ineffizient ist.

5. Quellenangaben

1. http://wiki.freepascal.org/Array/de
2. http://stackoverflow.com/questions/7417286/check-if-the-object-is-created-or-not-in-delphi
3. http://www.freepascal.org/docs-html/rtl/sysutils/getappconfigdir.html
4. https://www.delphi-treff.de/tipps-tricks/komponenten/allgemein/komponenten-zur-laufzeit-erzeugen/

Bild auf dem Cover: http://weknowyourdreams.com/images/snake/snake-01.jpg