***Computer Programming (CMP365, 366)/App and Game Development (CMP371)***

|  |  |
| --- | --- |
| CIP Code: | 110201 (intro/adv) / 110201 (games) |
| State Course Code: | 10154 (intro/adv) / 10197 (games) |
| Content Area Code: | 110 (all) |
| Instructor: | Mr. Miyoshi |
| Phone: | 425.844.4800 |
| E-Mail: | MiyoshiM@RSD407.org |
| Texts: | *C# with Miyoshi*, by Michael T. Miyoshi (intro)  *Fundamentals of Java*, by Lambert & Osborne (AP)  *Fundamentals of C++ and Data Structures*, by Lambert & Naps (adv)  *Tricks of the Windows Game Programming Gurus* by André LaMothe (games)  *Beginning DirectX 9* by Wendy Jones (games)  Other texts and supplements as needed |

Online Resources: <https://rsd407-my.sharepoint.com/:f:/g/personal/redwolf_rsd407_org/EgbH8Oo45mBJkG7Ib3mJxhwB2ROllDlGU60N2Le7_0K6Gg> [Note: You must use your rsd407 login credentials to get access to these resources.]

(No login credentials are needed to access the public GitHub resources listed below.)

<https://github.com/MichaelTMiyoshi>

<https://github.com/MichaelTMiyoshi/CSharpWithMiyoshi>

**Student Provided Materials and Class-use Fee**

Students should provide their own brains and enthusiasm. Students may download visual studio community or visual studio code (and Unity for Game & App Development students) from the internet so that they can program at home.

**Class Objective**

The objective of Computer Programming is to give the students exposure to, and to stimulate interest in the working environment of computer programming while providing the students with an opportunity to acquire beginning programming skills.

**Class Expectations**

Computer Programming is a year-long class for those who have an interest in computer science / computer programming or related fields. It is expected that the student who takes this class will obtain certain skills and knowledge important to those entering these fields. The skills and knowledge that the student will start to practice and learn are listed as competencies in a separate guide. These competencies will be demonstrated by the students through a variety of assignments (programs), projects, and tests.

**Attendance Policy**

Programming tools such as compilers are not at the disposal of every student nor are computers themselves. For this reason, most of the work will be done during the class periods. This makes class attendance extremely important. Students with an excused absence may make up the work for the class.

**GitHub**

GitHub is a professional tool used for source control and sharing projects. It is especially useful when changes are being made to projects both for archival purposes and for a history of what has been done. It is also useful for backing up projects. Students will also find it a handy tool when moving from computer to computer. It is recommended that all students get a GitHub account and use it to sync with their local projects. Students should make private repositories and they can make the instructor a collaborator on class-related repositories. It is suggested that GitHub Desktop be used to sync Unity projects (and architectural and engineering projects).

**Tardy Policy**

Roll will be taken and the school policy will be followed.

**Dress Code and Language**

The classroom should be considered the workplace for the students, and appropriate attire and language in the workplace is important. To help solidify these concepts, the Cedarcrest High School dress code will be observed at all times. Further, clothing with spaghetti straps or that show the midriff (belly) are specifically forbidden. Clothing that shows a student’s underwear is also forbidden. Clothing with drug, alcohol, gang, and/or hate related symbols or messages are also forbidden. Students will first be asked to change or cover the offending clothing described in the student handbook, above, and/or as deemed inappropriate by the instructor. Students who are continually asked to change or cover offending clothing will be marked tardy for the day when asked to change attire and are subject to the tardy policy. They will also be subject to penalties in the professionalism grade. Foul language is not permitted in class. Violators of the foul language policy are subject to the school disciplinary measures and the professionalism standards.

**Artificial Intelligence (AI)**

Artificial Intelligence is becoming an increasingly big (and arguably important) part of life these days. But it must be used with caution in education. As such, there is a new AI policy at Cedarcrest High School that will be adhered to in the programming classes as well.

AI is not necessarily easy to detect in computer programs. However, authorship is. Just as every writer has a unique style of writing, so it is that every programmer has a definite style of coding. Algorithms (how things are done) might be similar or even identical, but (except in the perhaps simplest cases) the implementation of an algorithm will be different depending on the programmer. Which is why students will be asked to explain their code, in comments and orally. When they are not able to explain their own code, something is amiss. Whether that something is AI or not can have far reaching consequences, especially since the school AI policy is cumulative through a student’s entire high school career. There is a big difference between getting 75% maximum (without the opportunity to redo it) because you cannot explain a project and having one of your AI strikes levied against you because you had AI write your code.

**Grades**

Grades will be determined by the following weighting of five categories:

1. **Professionalism (10%; Advanced 10%)** – Professionalism is an important part of the class. Aspects already mentioned such as dress, language, and attendance are important. As such, professionalism is evaluated each day. Four points are available each day. Any absence except school-related will receive no points until students make them up. Students who do not make up these professionalism points within a week are only eligible to make them up for 75% (3/4) of the points. Students who are tardy will receive 2 of the 4 points. Students who are disruptive, sleeping, playing games, swearing, etc. will receive 0 points and will not be allowed to make them up. All the above is to say that students who treat the class as a job ought to earn all of their professionalism points each day they are present.
2. **Leadership/Professional Development (15%; Advanced 15%)** – Each student will be given a list of activities in which he or she may participate for points. These activities range from leading the class in the flag salute to attending a national leadership conference. Students should also consider joining TSA, DECA, FBLA, WCTSMA, FFA, or other Career and Technical Student Organizations (CTSOs) for leadership and competition opportunities. Leadership points are accumulated throughout the school year.
3. **In-Class Assignments (20%; Advanced 0%)** – Students are given in-class assignments to help them master the content and skills of the class. These assignments may be in the form of tests (different from competency tests) or short programs that they will be required to finish in a short time period, sometimes by hand.
4. **Competencies (20%; Advanced 20%)** – As the students learn skills, they will be asked to perform practical timed tests. These tests are merely programs that must be completed in a class period at a certain skill level. Beginning students may retake competency tests, however, they may not take any test twice in the same day or on the day the test is scored. (Advanced students’ competencies are self-evaluated.)
5. **Project (35%; Advanced 55%)** – Students will create projects for much of their grades. This means that the application of the knowledge and skills they are gaining are what is most important. Beginning projects are done individually. Advanced students may create projects individually or in groups depending on the project. Students beyond the beginning level cannot earn an A grade for the class unless they complete at least one group project. The list of possible projects listed on GitHub is not comprehensive. Other project proposals must be submitted to the instructor before the student begins a proposed project. Advanced students are graded on their body of work. This body of work must be of high quality and sufficient quantity to receive high marks. This quality and quantity may be different for different students depending on the skill level they are currently at.

The weighted average of the above categories is taken and the aggregate score is assigned a grade as shown in the following chart.

|  |  |
| --- | --- |
| **Percentage** | **Grade** |
| 93.00 | A |
| 90.00 | A- |
| 87.00 | B+ |
| 83.00 | B (Students must earn a B or better to receive college credit.) |
| 80.00 | B- |
| 77.00 | C+ |
| 73.00 | C |
| 70.00 | C- |
| 67.00 | D+ |
| 60.00 | D |
| 0.00 | F |

Even though the class is a year-long course, the grade earned at the semester is a final grade and will not be changed if the student earns a higher grade in the second semester.

**Minimum Grades**

Minimum grades may be earned only in the competency category. If students have done work in the assignments and/or project sections that demonstrate skill in the required area(s), they will receive a minimum of 50% score for any competency test they have demonstrated the skill. Students receiving zeroes in any category (especially the competency category) are not doing enough relevant work to receive credit.

This minimum grade only applied to the competency category. All other categories accumulate points toward the students’ grades.

**Turn In Policy**

I believe all students can learn given enough time, but I also believe that students must be given realistic timeframes in which to operate. To help with each of these conflicting objectives, students will be held to certain times for turning in assignments. With few exceptions, all assignments can be turned in for full credit within two weeks of their dates in the gradebook. Competency tests must be taken within two weeks of the due dates in order for students to be eligible to retake the tests for full credit. If students have not taken a competency test after two weeks from the due dates, students may only take competency tests for a maximum of 75% of the possible points.

**Computers**

Computers are an important tool for the Computer Programming class and must be treated as such. Students who misuse computers especially by altering system files, introducing viruses, creating inappropriate or files will receive the appropriate sanctions as outlined in the school's computer use and internet agreement. All students enrolled in Computer Programming must have on file a signed district computer use agreement.

**Dual Credit – College Credit and Careers Network**

You can receive inexpensive Community/Technical College credit. To receive the credit, you must:

1. Be enrolled in a Dual Credit high school class (Intro, Game/App Dev)
2. Register on line for the Dual Credit program (https://www.ctesers.org/Home)
3. Pay the fee
4. Receive a B grade or higher!

**Competency Lists**

The following competencies are the skills that students are expected to gain when they take computer programming. The numbers come from different guiding agencies. Students in beginning and advanced classes will learn the competencies and some of the game programming competencies using the C# language. Those taking the AP Computer Science test will learn the Java programming language as well.

|  |
| --- |
| **Understands Terms and Procedures:**  **11102** Classify the different steps and the files created when making a new executable file.  **11103** Identify and use the different data types in this programming language to manipulate data.  **11106** Plan and design a program using structured development techniques such as algorithms and pseudocode. |
| **Understands and Uses Math and Logical Operators:**  **11104** Use mathematical operators, comparison operators, and logical operators to solve business problems.  **11105** Recall the correct level of precedence for each of those operators. |
| **Understands and Uses Decision Making Structures:**  **11107** Use if and if/else selection structures to decide which actions to perform in a program.  **11109** Use the switch statement when making multiple decision selections. |
| **Understands and Uses Repetition Structures:**  **11108** Use repetition structures (while, do-while, etc.) to repeat a group of statements. |
| **Understands and Uses Functions:**  **11110** Explain why functions are a necessary component of modularizing C++ programs.  **11111** Plan, design, create, and use functions in breaking down tasks to solve a problem.  **11112** Compare pass by value and pass by reference between functions. |
| **Understands and Uses Arrays of Different Types:**  **11113** Write valid programming statements to declare and initialize arrays, to refer to individual elements of an array, and pass arrays to functions.  **11114** Use and manipulate strings of characters, including the Standard Library string class. |
| **Understands and Uses Complex Data Types:**  Specify, define, implement, and use structures, enumerated and other user defined data types.  **11117** Specify, define, implement, and use simple classes. |
| **Design:**  **B1** Develop detail design specifications. |
| **Development:**  **C1** Create and modify new or existing system interfaces.  **C2** Create and modify new or existing code. |
| **Project Management:**  **F1** Define scope of project.  **F4** Estimate time requirements.  **F7** Evaluate project requirements. |
| **Debugging:**  **11116** Use the debugging tools available with the current compilers. |
| **Understands and Uses Complex and Dynamic Data Structures:**  Create and use dynamic data structures including but not limited to linked lists, stacks, queues, and trees.  **11101** Create and use pointers and dynamic memory allocation. |
| **Understands and Uses Recursion:**  Create and use recursion to solve problems. |
| **Problem Solving:**  **11115** Define the general scope of work to meet project requirements or solve a problem.  **H1** Define the problem.  **H3** Identify/test possible solutions.  **H5** Implement solution. |
| **Analysis:**  **A1** Be able to gather data to identify customer requirements.  **A2** Interpret and evaluate requirements.  **A3** Define scope of the work to meet customer requirements.  **A4** Develop high level systems and functional specifications. |

In addition to the above competencies, Game Programming students have the following competencies.

|  |
| --- |
| **Input:** The student demonstrates skill in receiving input from appropriate devices including but not limited to the keyboard and mouse. |
| **User Interface:** The student demonstrates skill in designing user interfaces that are easy and efficient to use. May include (but not limited to) menus, Heads Up Display (HUD), and other indicators. |
| **Surfaces/Sprites:** The student demonstrates skill in making surfaces/sprites for bitmaps and manipulates them according to the structure of the game/simulation. This includes background scrolling and manipulating onscreen and offscreen surfaces/sprites. |
| **Sprites/Animations:** The student demonstrates skill in manipulating surfaces/sprites to form animations that are applicable and appropriate to the game/simulation. |
| **Collision Detection:** The student demonstrates an understanding of and effectively uses collision detection as appropriate to the game/simulation. Including object to object and object to cursor. |
| **Physics:** The student demonstrates skill in manipulating the physics of the environment according to desired parameters. |
| **Automation:** The student demonstrates skill in automating objects according to the requirements of the game/simulation. This includes but is not limited to random events. |
| **Sound:** The student demonstrates skill in creating and manipulating sound in the game/simulation. This includes but is not limited to tone generation and playing wave and midi files. |
| **3D:** The student demonstrates skill in creating and manipulating objects in a 3D environment. |
| **Artificial Intelligence:** The student demonstrates skill in creating artificial intelligence relevant to the game/simulation. |
| **Debugging:** Student uses various devices to debug programs. This includes but is not limited to outputting variables and using the compiler’s debugger. |