***Mass transfer between roots and shoot (between 2dsoil and crop.dll)***

2DSOIL variables:

*PCRS*:Total amount of carbon used to grow roots, g Carbohydrate day-1

*PCRL* Minimum carbon needed to grow roots g Carbohydrate day-1

*PCRQ* Maximum carbon provided to grow roots g Carbohydrate day-1

Crop variables:

*rootPart* mass of carbohydrate partitioned to roots in current time step, plant class

*rootPart\_Real* actual amount of carbohydrate partitioned to roots in the previous time step, plant class.

*mass* total mass gr of carbohydrate in root, organ class (inherited by root class) (per plant)

Note that *PCRQ* is set as the total photosynthesis for the current hour. If the plant needs more than the minimum (*PCRL*) in case of water stress, some, but not necessarily all of *PCRQ* will be used to grow roots. The rate of actual carbon use is in the variable *PCRS*.

In the crop model, *PCRL* and *PCRS* are sent to 2DSOIL as:

SHOOTR->PCRL=(pSC->getPlant()->get\_rootPart())\*24\*PopSlab;

SHOOTR->PCRQ=(pSC->getPlant()->get\_rootPart())\*24\*PopSlab;

24 scales from hour to day, and *popslab* scales from plant to slab*. RootPart* is the grams of **carbohydrate** partitioned to the root in one hour.

Note: this code in plant.cpp (about line 241)

The statement (**C\_pool += assimilate\*CH2O\_MW/CO2\_MW;** // convert from grams CO2 to grams carbohydrate)

assigns the assimilate value from photosynthesis to a variable holding carbohydrate*. C\_Pool* is the source of carbon partitioned to other plant parts.

Here *C\_pool* is allocated to *C\_Supply* based on a temperature effect from Robert Grant's paper on partitioning. This is one of several methods to transfer carbon to *C\_supply*. The method varies based on reserves and demand. See line 486 or so in plant.cpp. See code for description of *grofac* and *tempEffect*. The method **C\_allocation** contains the code.

**C\_supply = \_\_max(C\_pool\*tmprEffect\*grofac, 0);** //CADD from Grant

Next C\_supply is partitioned to the roots as grams of carbohydrate

**rootPart = \_\_max(0,Yg\*((1-Fraction)\*(C\_supply-maintRespiration)));** // gCH2O partitioned to roots

Next the carbohydrate is sent to 2DSOIL:  
**SHOOTR->PCRL=(pSC->getPlant()->get\_rootPart())\*24\*PopSlab;**

**SHOOTR->PCRQ=(pSC->getPlant()->get\_rootPart()+ pSC->getPlant()->get\_shootPart())\*24\*PopSlab;**

After processing by 2DSOIL, the total carbohydrate amount used is returned to the plant:

**wthr.pcrs = SHOOTR->PCRS/PopSlab/24;**

Logic for root growth and carbon. We found that the carbon allocated in one time step does not get used in that time step because of low sink, Therefore, we had to spread the carbon allocation over 24 hours and not just when the plant was photosynthesizing. When carbon is allocated to the roots, the allocation amount for the previous time step is stored in rootPart\_old. The actual amount of carbon used in the last time step for root growth is stored in w.pcrs. This value may be

1. less than the amount allocated if the sink was too low (w.pcrs>rootpart\_old)
2. Greater than the amount allocated if carbon was needed to grow extra roots
3. Equal to the amount allocated if root growth proceeded normally.

The storage location of the returned *pcrs* is confusing now because the structure wthr is only used once. I will make the returned value of *pcrs* a member variable of the plant class. A new structure **'w'** is used later to access the variable. The difference is because the variable name is used as an argument in method calls. The structure itself is defined in weather.h and is used to transfer environmental variables when calling different methods. In that sense *pcrs* does not belong. I have to get rid of the variable name 'w' and rename the structure something other than 'weather', If you notice, it is called **TWeather** in weather.h and this is because it comes from Soo's original Pascal program he wrote for the photosynthesis routine. I translated it but kept the original variable names. Any variables declared as type **TWeather** in calling statements are of this type.

*w.pcrs* (the total actually used) is compared to *rootPart* from the past time step (*rootPart\_old*) to see if additional carbon was used. If it is greater than *rootPart* then the difference is subtracted from the total carbohydrate available to the shoot for the next time step.

The variable *rootPart*\_real is updated from *rootPart* to account for the additional carbohydrate used (over the minimum supplied) in the method **C\_allocation** in plant.cpp (line 563).

The root mass in the plant is updated as (in plant.cpp, **C\_allocation** method ,about line 756):

**if (develop->Emerged()) this->get\_roots()->import\_CH2O(rootPart\_real);**

The method **import\_CH2O** increments *dCH2O* and sets *mass* to *CH2O*. This is a virtual method in the **organ** class

Two or three lines above we have:

**this->get\_roots()->set\_ActualCarboIncrement(rootPart\_real);**

The method **ActualCarboIncrement** is a virtual function in the **organ** class. It saves the increment of actual carbohydrate available in the past time step. The variable is used to compare to potential carbohydrate available in order to calculate stress effects. It is probably not relevant for root growth at this time.

There is a method called **setMass** in plant. This sets the mass for the plant. It has no arguments and calls the **get\_mass** routines for the different organs. This method sets the variable *rootMass* which is a member for plant by calling the method **get\_Mass** in the root class.

Overall:

The value of root mass is incremented and stored in the **root** class (as member variable *mass*) after *pcrs* is returned from 2DSOIL. Each time step the plant class keeps a tally of the masses of the different components using the method **setMass**. The plant object gets the root mass by calling a virtual function in the root class called **get\_mass**. When the values of the variables are output into the file, only the plant class needs to be called in order to obtain a sum of the component masses and the mass of each component. If, at some point we have different classes of roots each with their own mass, we can keep track of them in the root class. The **plant** class only cares about totals for each organ. The root mass is stored in the variable *rootMass* in the plant class.